1581 DOS REFERENCE GUIDE

©1988 David W. Martin
Published by HOS Inc.

First Edition, October 1, 1988

Published by Hands on Software, Inc.

Copyright 1988 David W. Martin

All Rights Reserved.

2700 NE Andresen Road #A-1
Vancouver, WA, USA 98661
(206)695-9648

This book is copyrighted. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise.

Every effort has been made to insure complete and accurate information concerning the material presented in this book. However, David W. Martin can neither guarantee nor be held legally responsible for any mistakes in printing or faulty instructions contained in this book. The author will always appreciate receiving notice of any mistakes.

1541, 1571, 1581, Commodore 64, Commodore 128, BASIC 7.0 , BASIC 2.0 and Amiga are trademarks or registered trademarks of Commodore Int., Ltd.

CP/M is a registered trademark of Digital Research, Inc.

This book is dedicated to

my family

on both sides of the Atlantic.

* * *

Dieses Buch ist gewidnet zu

meine Familie

an beiden seiten von den Atlantik.

Table of Contents

Introduction.....i

Chapter 1 - New Disk Drive Commands.....1

1581 Extended Block Read/Write.....1

1581 Utility Loader - "&".....1

1581 Autoloader.....2

1581 Disk Partitioning.....3

View 1581 "Secret messages".....6

Chapter 2 - Disk Format Structure.....7

Format Organization.....7

Track and Sector Layout.....8

Physical to Logical Conversions.....8

Directory Track.....11

BAM - Block Availability Map.....12

File Information Sectors.....14

Directory/File Entries.....15

Program File Storage.....16

Sequential/User File Storage.....17

Relative File Storage.....18

Deleted File Storage.....19

Locked File Storage.....19A

Chapter 3 - Using Burst Commands.....20

Read.....21

Write.....22

Inquire Disk.....23

Format Disk.....23

© Copyright 1988 D. W. Martin All Rights Reserved.

Query Disk Format.....25
Inquire Status.....26
Dump Track Cache Buffer.....27
Fastload Utility.....28
Burst Status Byte Decoded.....28
Burst Routine Notes.....29
Chapter 4 - Burst Utility Commands.....31
Serial Bus Mode.....31
DOS Sector Interleave.....31
DOS Retries.....32
ROM Signature Analysis.....32
Verify Select.....32
Disk Device Set.....33
Mystery Command - SIEEE Timing.....33
Memory Read/Write.....34
Chapter 5 - Using Kracker-Mon.....35
Chapter 6 - System Overview & RAM Memory Map..41
System Overview Map.....41
RAM Memory Map.....42
Chapter 7 - DOS Controller Chips Memory Map...57
8520A 2 Mhz CIA Input/Output Controller.....57
WD177x Floppy Disk Controller.....58
WD177x Command Summary.....58
WD177x Status Byte.....60
Programming the WD177x.....61
Chapter 8 - ROM Memory Map.....67

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 9 - Miscellaneous.....107

Powerup Diagnostics.....107

Track Cache Buffer.....107

1581 Serial Bus Specifications.....108

Serial Bus Pin Out.....109

Serial Bus Commands/Addresses.....110

Secondary Address Descriptions.....111

Standard Kernal Calls & Serial Bus.....111

Standard Serial Bus Protocol.....114

Fast Serial Bus Protocol.....114

Burst Serial Bus Protocol.....115

1581 Bugs.....115

Reaching the Author.....117

Getting Information on the WD177x.....117

Recommended Reading.....118

Construction - Building a Book.....119

Thanks.....119

Appendix A - Sample and Utility Programs.....121

Loading Sample and Utility Programs.....121

Loading Kracker-Mon.....121

Sample Program Documentation.....122

Appendix B - 1581 Error Creator.....125

Appendix C - Programming the 1581 Job Queue...127

The 1581 Job Queue.....127

Job Queue Programmer's Memory Map.....128

Basic 1581 Controller Call.....128

© Copyright 1988 D. W. Martin All Rights Reserved.

1581 Controller Status Byte.....130

Job Contoller Commands and Descriptions....130

DOS Kernal Jump Table.....145

© Copyright 1988 D. W. Martin All Rights Reserved.

First, I'd like to point out that this book is not a substitute for the 1581 manual provided by Commodore. So don't throw that book away yet. My book is designed to compliment the Commodore book that in many ways is a step ahead of previous Commodore disk drive manuals. They actually provided information on the DOS in the 1581 book. I was really impressed by it, but the need to learn and know more gave me the desire to write the book you are reading: The 1581 DOS Reference Guide.

I'm sure that many of you share the same feelings as I do. It is always best to know as much about the equipment you use that's possible. That way you can get that extra ounce of performance out of it, etc. Well this books goal is to give you that information the rest is up to you. So get cracking! Let's see some really hot 1581 disk utilities.

Speaking of hot utilities, the folks at Software Support International have done their part in creating the best 1581 utility package around. In fact, you'll find the hottest 1581 utility around included on a disk with this book. It's one software package that no 1581 owner should be without. It will catch your 1581's real spirit and let it fly high!

The second thing I'd like to mention is about the memory maps in Chapters 6 through 8. The maps provided do not contain any of the original program code contained within the 1581's DOS ROMs. The reasons behind this are many, but basically the main reason is the fact that Commodore now includes a copyright message within the DOS ROMs that would hamper publication of an extensive memory map. Therefore, I am only providing routine addresses, labels and descriptions in the DOS ROM memory map.

The third thing I'd like to mention is that this book is not going to teach beginners to program the 1581 DOS. It's contents are provided as a reference to experienced computer programmers, DOS programmers and beginners interested in what makes the 1581 tick. If you are new to DOS programming I suggest that you pick up any of the books mentioned in the recommended reading section in chapter 9. The books listed there will help those of you who are interested in programming other Commodore or compatible drives (i.e. 1541, 1571, and MSD drives).

In closing I'd like to say thanks for buying this book. The book has been a product of many man hours plus the usual blood sweat and tears. Please

© Copyright 1988 D. W. Martin All Rights Reserved.

encourage your friends to buy this book and it's companion disk. Do not pirate these products. It only hurts me and you in the end. Remember! Just say NO to piracy!

David W. Martin

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 1 - New Disk Drive Commands

1581 Extended Block Read and Write

The 1581's extended BLOCK-READ and BLOCK-WRITE commands allow you to read tracks and sectors that are beyond the normal range used by the DOS. They function in a manner that is similar to the U1 and U2 commands, except the ILLEGAL TRACK AND SECTOR message is bypassed. As in the U1 and U2 commands the buffer pointer is initialized to point at the first byte in the sector read.

I have not found a use for this feature yet, but it may possibly be useful in disk protection. For instance, the Amiga's disk drives can read out past track 80 and data can be stored on these extended tracks. It maybe possible in the near future to format 1581 disks out past track 80 and then use the extended B-R and B-W commands to retrieve and store data there. These commands take the following format:

EXTENDED BLOCK-READ

```
PRINT#15,"B-x";channel#;track#;sector#
(x = shifted R)
                    or
PRINT#15,"B-";CHR$(210);channel#;track#;sector#
```

* * *

EXTENDED BLOCK-WRITE

```
PRINT#15,"B-x";channel#;track#;sector#
     (x = shifted W)
                    or
PRINT#15,"B-";CHR$(215);channel#;track#;sector#
```

* * *

1581 UTILITY LOADER - "&"

The 1581's Utility Loader (UL) command is a handy feature that can be used to load and execute machine language DOS programs inside the 1581. It's command structure is as follows:

```
PRINT#15,"&0:filename"
```

© Copyright 1988 D. W. Martin All Rights Reserved.

The file loaded by this command must be in the following format:

```
Byte 0 - Low byte of start address

Byte 1 - High byte of start address

Byte 2 - Number of program bytes

Byte 3 - Data bytes containing program code
         .                          .
         .                          .
         .                          .
Byte x -                            .

Byte x+1 - Checksum
```

The file type must be of type USR so the UL will function properly and you are limited to a program size of 251 bytes. Program data bytes MUST start at byte three.

The start address also doubles as the load address of the program in the disk drive's RAM and execution will begin at that point. Upon execution of an RTS instruction control will return to the user.

The checksum is always the byte following the last data byte. It is calculated by adding the starting address, the number of data bytes and the data bytes themselves together. If this calculation results in a carry it is counted along with the checksum.

Appendix A contains some demo programs that demonstrate the usage of the UL command. The **PRG-2-USR** program will take your machine language program file on disk and convert it to a USR file that is compatible with the UL command. The **DEMO & CMD** program loads a USR file on disk that demonstrates the "&" command by flashing the disk drive error light before returning control of the disk drive to the user.

* * *

1581 AUTO LOADER

The 1581's auto loader is a handy feature for executing a DOS program automatically when the drive performs any of the following functions: power-up, reset, burst inquire, burst query, or the initialize command. During these operations the 1581 looks for a

© Copyright 1988 D. W. Martin All Rights Reserved.

file called "COPYRIGHT CBM 86" that is a USR type file. This file must follow the same format used in the utility loader mentioned previously. The file will be loaded and executed if it is present on the disk in the drive.

The automatic loading feature can be disabled in the following ways:

(1) Renaming the file (easiest)
(2) Setting a flag in the BAM
(3) Setting a flag in DOS RAM

After executing the program must call the JCBMBOOTRTN ($FFBA) to return control to the 1581.

Appendix A contains a file that will demonstrate the 1581's auto loading file capability. The **AUTOBOOT DEMO** program will cause the 1581's error LED to flash as in previous examples.

Notes:

1) See location $01FB in the RAM memory map. Bits 7 and 6 control the autoboot sequence.

2) See byte number 7 in the BAM (track 40 sectors 1 and 2) it controls the autoboot sequence also.

* * *

1581 DISK PARTITIONING

The 1581 supports disk partitioning via the "/" command. Disk partitioning provides a means of protecting portions of the diskette. This was provided so that users could permanently reserve space on a diskette for BOOT sectors, CP/M data, or random access files.

Reserving disk space has always been easy via the BLOCK-ALLOCATE command (B-A). The B-A command would set a bit in the BAM that would protect the sector from being over written by normal DOS commands. The BLOCK-FREE (B-F) command could then be used to deallocate a selected disk sector.

Although easy to use the B-A command is not without problems. It's main problem has always been the fact that sectors that are allocated by it can be de-allocated by the DOS VALIDATE command.

This is where the disk partition command steps

© Copyright 1988 D. W. Martin All Rights Reserved.

in. It prevents the deallocation of special blocks by the VALIDATE command. This is because the VALIDATE command skips partition files that are designated as file type CBM.

CBM files are created when you set up a partition and they insure the protection of the selected area on the diskette. As long as the CBM file exists the data will be protected from the VALIDATE command.

Partitions are named the same way as other disk files on the 1581. Like them you are limited to sixteen characters, etc. When listing a directory they appear as file type CBM.

The create a partition command is as follows:

```
PRINT#15,"/0:partition name,"+CHR$(starting
 track)+CHR$(starting sector)+CHR$(low byte
 of sectors)+CHR$(high byte of sectors)+",C"
```

Larger partitions can be used as sub-directories. There are, however, some limitations if you plan on using parts of a partition as a sub-directory. They are:

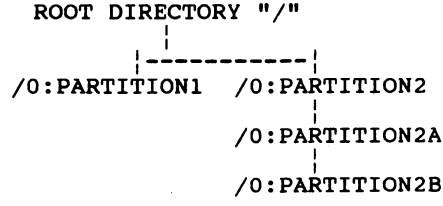
(1) The partition area must be at least 120 sectors in size

(2) The starting sector must always be 0

(3) The ending sector must always be a multiple of 40

(4) The area allocated cannot contain track 40 (which is the original system track for the root directory)

Many levels of sub-directories can be created. The figure below shows a graphical representation of multiple level disk partitions.



Once disk partitions are created they can be

© Copyright 1988 D. W. Martin All Rights Reserved.

selected via the command:

PRINT#15,"/0:partition name"

The first time you open a disk partition you must format it via the DOS NEW command. Care should be taken that the selected partition is available or you may end up formatting the entire diskette. You can check to see that you have selected the right partition via the error channel. If everything is ok then reading the error channel should produce:

02, SELECTED PARTITION,first track#,last track#

If the error channel produces:

77, SELECTED PARTITION ILLEGAL,00,00

then you must reattempt to select the partition again. It either does not exist or you misspelled the file name.

Directories can only be accessed in a forward direction. If you need to go back to a directory a level above the current one, you must first select the root directory via:

PRINT#15,"/"

Then you will need to work your way down through the other directories until you reach the level you need. The example below shows you how to switch between multiple level directories.

Example: Multiple Directory Switching

```
PRINT#15,"/":PRINT#15,"/0:PARTITION2":
PRINT#15,"/0:PARTITION2A"
```

(Places the current directory at PARTITION2A)

PARTITION NOTES

The selected partition becomes the default working area. All accesses to the disk drive will only be done in this area. Files outside this area are invisible to you as if they do not exist.

The first track of a disk partition acts like track 40 in the root directory. It is your responsibility to see that no information over writes this track or corrupts the data on it. The DOS only protects track 40 of the root directory.

© Copyright 1988 D. W. Martin All Rights Reserved.

Removal of partitioned areas is easy. Simply SCRATCHING the partition file entry in the appropriate area will free up the allocated disk space. Files in the partition will be lost, so be sure to copy the files that you might need later.

* * *

VIEW 1581 "SECRET" MESSAGES

The 1581 disk drive supports two commands not listed in the manual supplied with the 1581 they are the "B-*" command and the "B-?" command.

The "B-*" command is the dedication message placed in the DOS by one of it's designers. Sending this command via the command channel and then reading that channel generates the message:

"7:,DEDICATED TO MY WIFE LISA,00,00"

Notice the odd error message number. It represents error number $7A in the DOS.

The "B-?" command is the author message. It tells you who designed the 1581 hardware and software. If you want to see this message send it via the command channel and then read that channel it will generate the following message:

"79,SOFTWARE DAVID SIRACUSA. HARDWARE GREG BERLIN"

Hidden messages seem to be a recurring item in most new Commodore products. The C128D is the first piece of equipment where I encountered this novelty. If you want to see the C128's hidden message simply type the following in C128 mode:

SYS32800,123,45,6 (Hit RETURN)

The 1571 disk drive also contains a message near the beginning of it's operating system ROM ($8000). Currently, I am not aware of a DOS command that will cause it to be printed via the command/error channel.

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 2 - Disk Format Structure

1581 FORMAT ORGANIZATION

Gross Data Organization:

3.5 inch disk
Double-sided
80 cylinders/160 tracks

Physical Format Organization:

Cylinders 0 thru 79
Sectors 1 thru 10 on side 0
Sectors 1 thru 10 on side 1
Sector size 512 bytes

Logical Format Organization:

Tracks 1 thru 80
Sectors 0 thru 39 (using physical sectors 1 to 10 sides 0 and 1)
Sector size 256 bytes

Storage:

Total unformatted capacity	1 Megabyte
Total formatted capacity	808,960 bytes
Maximum sequential file size	802,640 bytes
Maximum relative file size	approx. 800K
Records per file	65535
Files per diskette	296*
Cylinders per diskette	80
Logical sectors per cylinder	40 (0 to 39)
Physical sectors per cylinder	20 (1 to 20)
Free blocks per disk	3160
Logical bytes per sector	256
Physical bytes per sector	512

* More with disk partitioning (each partition can hold another 296 files)

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

1581 TRACK AND SECTOR LAYOUT

1581 per track organization:

Hex 4E written as a gap, with 10 sectors of data, with full gaps written for motor speed variation.

1581 per sector organization:

MFM encoded contents of each sector:

12 bytes of $00
3 bytes of hex $A1 (data hex $A1, clock hex $0A)
1 byte of $FE (ID address mark)
1 byte (track number)
1 byte (side number)
1 byte (sector number)
1 byte (sector length, $02 for 512 byte sectors)
2 bytes CRC (cyclic redundancy check)
22 bytes of hex $4E
12 bytes of $00
3 bytes of hex $A1 (data hex $A1, clock hex $0A)
1 byte of hex $FB (data address mark)
512 bytes of data
2 bytes CRC
38 bytes of hex $4E

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

1581 PHYSICAL TO LOGICAL FORMAT CONVERSION

CYLINDERS VS. TRACKS

The 1581's physical cylinders range from 0 to 79 on the disk surface, whereas logical tracks range from 1 to 80.

To convert from cylinders to tracks use the following formula:

CYLINDER # + 1 = TRACK #

To convert from tracks to cylinders use the following formula:

TRACK # - 1 = CYLINDER #

* * *

Sector Conversion

Physical to logical sector conversions are where we approach the hard part of the conversion process. We actually have to do a little bit of math.

The 1581's physical sectors occupy sectors 1 thru 10 on sides 0 and 1 of the diskette and contain 512 bytes of information per sector. The equivalent logical sectors can be computed using the following formulas:

LOGICAL SECTOR A = (PHYSICAL SECTOR# * 2) - 2 + (20 * SIDE#)

LOGICAL SECTOR B = (PHYSICAL SECTOR# * 2) - 1 + (20 * SIDE#)

NOTE: Since a physical sector contains two logical sectors you must remember to work on each part of the 512 byte block as two 256 byte blocks. Logical sector A is equivalent to: START OF 512 BYTE DATA BLOCK + 0 and logical sector B is equivalent to: START OF 512 BYTE DATA BLOCK + 256.

The 1581's logical sectors occupy sectors 0 thru 39 on sides 0 and 1 of the diskette and contain 256 bytes of information per sector. The equivalent physical sectors can be computed using the following formulas:

© Copyright 1988 D. W. Martin All Rights Reserved.

SIDE 0 - LOGICAL SECTORS 0 TO 19

PHYSICAL SECTOR = (LOGICAL SECTOR EVEN# + 1) / 2
+ .5 + (20 * SIDE#)

PHYSICAL SECTOR = (LOGICAL SECTOR ODD# + 2) / 2
- .5 + (20 * SIDE#)

NOTE: SIDE# = 0 in the two formulas above.

SIDE 1 - LOGICAL SECTORS 20 TO 39

PHYSICAL SECTOR = (LOGICAL SECTOR EVEN# + 2) / 2
- .5 - (10 * SIDE#)

PHYSICAL SECTOR = (LOGICAL SECTOR ODD# + 1) / 2
- (10 * SIDE#)

NOTE: SIDE# = 1 in the two formulas above.

The following chart may also be useful as a quick reference for physical to logical sector conversions.

SIDE 0		SIDE 1	
PHYSICAL	LOGICAL	PHYSICAL	LOGICAL
1	0, 1	1	20, 21
2	2, 3	2	22, 23
3	4, 5	3	24, 25
4	6, 7	4	26, 27
5	8, 9	5	28, 29
6	10, 11	6	30, 31
7	12, 13	7	32, 33
8	14, 15	8	34, 35
9	16, 17	9	36, 37
10	18, 19	10	38, 39

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Directory Track

The 1581's directory track is located on logical track 40 and is 40 disk sectors in length. The DOS uses the directory to manage up to 296 files per diskette.

The directory is composed of 40 disk blocks the first three have special functions (sectors 0, 1, and 2) the last 37 are used for file storage information (8 files per sector, 8 * 37 = 296).

Below are charts showing you the contents of sectors 0, 1, and 2 which are the directory header, BAM one, and BAM two respectively.

Directory Header (Track 40, Sector 0)

Byte #	Description
0	Track containing next directory block (normally 40/$28)
1	Sector containing next directory block (normally 3/$03)
2	Disk version number (normally 68/$44 ASCII "D")
3	Null byte (0/$00)
4-19	Disk Name (16 characters)
20-21	Shifted spaces (160/$A0)
22-23	Disk ID (2 characters)
24	Shifted space (160/$A0)
25	DOS version number (normally 51/$33 ASCII "3")
26	Disk version number (normally 68/$44 ASCII "D")
27-28	Shifted spaces (160/$A0)
29-255	Null bytes (0/$00)

NOTE: Locations 29 to 255 are not used by the DOS a variety of things can be stored here. Examples

© Copyright 1988 D. W. Martin All Rights Reserved.

are: text data, DOS programs, messages, etc. Use your imagination. Also, changing byte 3 to anything other than 68/$44 will "soft-write-protect" a diskette.

* * *

BAM - Block Availability Map

The BAM is located on sectors 1 and 2 on logical track 40 of a 1581 diskette. The BAM is used to keep track of blocks on the disk that are either reserved or available.

BAM 1 - Logical Tracks 1-40 (Track 40, Sector 1)

Byte #	Description
0	Track containing next BAM block (normally 40/$28)
1	Sector containing next BAM block (normally 2/$02)
2	Disk version number (normally 68/$44 ASCII "D")
3	Compliment version number (normally 187/$BB)
4-5	Disk ID (2 characters)
6	Input/Output byte Bit 7 Verify on/off Bit 6 Check header CRC on/off (normally 192/$C0)
7	Auto loader flag (normally 0/$00)
8-15	Null Bytes (normally 00/$00)
16-255	BAM image (logical tracks 1 to 40 using 6 bytes per track)

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

BAM 2 - Logical Tracks 41-80 (Track 40, Sector 2)

Byte #	Description
0	Track containing next BAM block (normally 0/$00)
1	Sector containing next BAM block (normally 255/$FF)
2	Disk version number (normally 68/$44 ASCII "D")
3	Compliment version number (normally 187/$BB)
4-5	Disk ID (2 characters)
6	Input/Output byte Bit 7 Verify on/off Bit 6 Check header CRC on/off (normally 192/$C0)
7	Auto loader flag (normally 0/$00)
8-15	Null Bytes (normally 00/$00)
16-255	BAM image (logical tracks 41 to 80 using 6 bytes per track)

NOTE: Bytes 2 thru 7 are copies of the bytes appearing in BAM 1 (Track 40, Sector 1).

* * *

BAM Image Interpretation

The bytes containing the BAM image are used as follows:

Byte #	Description
0	Number of free sectors on the track
1	Each bit marks a sector (sectors 0 to 7)
2	Each bit marks a sector (sectors 8 to 15)

© Copyright 1988 D. W. Martin All Rights Reserved.

Byte #	Description
3	Each bit marks a sector (sectors 16 to 23)
4	Each bit marks a sector (sectors 24 to 31)
5	Each bit marks a sector (sectors 32 to 39)

A sector is free if it's representative bit equals one. If the bit value is zero then the sector is reserved.

* * *

File Information Sectors (Track 40, Sectors 3-39)

Byte #	Description
0	Track of next directory block (normally 40/$28)
1	Sector of next directory block (normally 4/$04 and incrementing by one on each of the following directory sectors)
2-31	File entry number 1
34-63	File entry number 2
66-95	File entry number 3
98-127	File entry number 4
130-159	File entry number 5
162-191	File entry number 6
194-223	File entry number 7
226-255	File entry number 8

NOTE: There are two bytes between each file entry they contain a null value (0/$00). Thirty bytes are reserved for each file or directory entry (see next page).

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Directory/File Entry Structure

Byte #	Description
0	File types: 0 = Deleted file (DEL) 1 = Sequential file (SEQ) 2 = Program file (PRG) 3 = User file (USR) 4 = Relative file (REL) 5 = CBM file (CBM) If the file type is OR'ed with 128/$80 then it is a properly closed file. If the file type is OR'ed with 192/$C0 then it is a locked file and cannot be scratched.
1	Track of first data block
2	Sector of first data block
3-18	File name padded with shifted spaces (160/$A0) 16 characters in length
19	For relative (REL) files only! Track of the super side sector block.
20	For relative (REL) files only! Sector of the super side sector block.
21	For relative (REL) files only! Record length.
22-25	Null bytes (normally 0/$00)
26	Track number of replacement file during a SAVE or OPEN with replace "@"

© Copyright 1988 D. W. Martin All Rights Reserved.

Byte #	Description
27	Sector number of replacement file during a SAVE or OPEN with replace "@"
28	Number of blocks in the file (low byte)
29	Number of blocks in the file (high byte)

* * *

Program File Storage

Sector 1

Byte #	Description
0	Track of next block in the program
1	Sector of next block in the program
2	Program load address (low byte)
3	Program load address (high byte)
4-255	Program information stored on disk

Remaining Full Program Sectors

Byte #	Description
0	Track of next block in the program
1	Sector of next block in the program
2-255	Program information stored on disk

© Copyright 1988 D. W. Martin All Rights Reserved.

Last Program Sector

Byte #	Description
0	Null byte (normally 0/$00)
1	Number of valid data bytes in this sector
2-xxx	Last bytes of program information

* * *

Sequential/User File Storage

Byte #	Description
0	Track of the next sequential data block
1	Sector of the next sequential data block
2-255	Data bytes (254)

NOTE: The last sector for the sequential/user files above is the same as the one for program files.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Relative File Storage

Relative File Data Blocks

Byte #	Description
0	Track of next data block
1	Sector of next data block
2-255	Data bytes. Normally relative files records are empty if the first record byte is 255/$FF. The remaining bytes are filled with nulls (0/$00). Records that are partially filled are padded with nulls.

Relative File Side Sectors

Byte #	Description
0	Track of next side sector block in the current group
1	Sector of next side sector block in the current group
2	Side sector number (0 to 5)
3	Record length
4-5	Track and sector of side sector 0
6-7	Track and sector of side sector 1
8-9	Track and sector of side sector 2
10-11	Track and sector of side sector 3
12-13	Track and sector of side sector 4
14-15	Track and sector of side sector 5
16-255	Track and sector pointers to 120 data blocks

© Copyright 1988 D. W. Martin All Rights Reserved.

Relative File Super Side Sectors

Byte #	Description
0	Track of first side sector group 0
1	Sector of first side sector group 0
2	Value 254/$FE
3-4	Track and sector of first side sector in group 0
5-6	Track and sector of first side sector in group 1
.	.
.	.
.	.
253-254	Track and sector of first side sector in group 125

NOTE: Each super side sector has pointers to groups of 126 side sectors. Each side sector group contains 6 side sectors. Now, each side sector block points to 120 data blocks, each of which contain 254 data bytes. Thus, the maximum relative file size is 23,042,880 bytes (126 * 6 * 120 * 254 = 23,042,880).

* * *

Deleted File Storage

Deleted files on a diskette are represented by DEL when you list a disk directory. They can be manipulated like over files via LOAD and OPEN commands. However, when loading a DEL file type you must use the following LOAD command:

```
LOAD "PRGNAME,DEL,R",8
```

or

```
OPEN 8,8,8,"PRGNAME,DEL,R"
```

NOTE: DEL maybe abbreviated to just "D. "

Experiment with the various commands for loading and saving data and see what results you get. The commands above are also useful on program, sequential, and user files.

© Copyright 1988 D. W. Martin All Rights Reserved.

Locked File Storage

Locked files are files that cannot be scratched from the diskette. In order to scratch a locked file you must first unlock it. When listing a directory, a locked file will have a less than ("<") symbol appended after the file type.

The DOS does not currently have a command that will set the lock bits on the file type. You must either set the bit by hand using a sector editor or you can use a variety of public domain programs that will let you choose a file name to lock.

A program that will lock and unlock files on the 1581 is provided on the companion disk. See Appendix A for more information on **81FILELOCKON.OFF** program.

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 3 - Using Burst Commands

Commodore 128 owners will be happy to discover that the 1581 supports the burst protocol that was first introduced in the 1571. The protocol provides fast serial transfers of data and programs between the disk drive and computer.

The Commodore 128 computer is currently the only computer that supports the burst protocol. The C128's enhanced serial communication hardware and 2 Mhz clock speed make this possible. The C64's hardware cannot support the burst protocol since it lacks the C128's enhancements.

Unfortunately, the hardware is not the only limitation involving the burst protocol. Another is software support. BASIC cannot properly handle burst transfers since it is to slow. So in order to properly access burst transfers a programmer must turn to machine language.

Fortunately, Commodore has provided machine language burst routines on the 1581 demo disk that run on the C128. They provide the ML routines, their source code and some sample programs. The sample programs show you how to use their ML burst routines with BASIC.

The source code files are provided in the CBM Macro Assembler format. You can use these files in other assemblers with some changes. However, one problem you may encounter is that some or all of the demo disks contain a defective source code file (if you find a good one let me know). You may need to make a new source file (using the supplied object file) with a disassembler.

The list file is intact, but it is not useful as a file for an assembler. However, it's very useful as a burst routine reference and it contains the documentation for the burst routines. I recommend that you print out a listing of this file and that if you do you should print it out in 132 columns on your printer. Note, that if you do not set your printer to print in 132 column compressed print the listing will be hard to read.

The following pages contain information on using the burst routines on the 1581. The burst command syntax is provided as a reference in a form that I hope is more understandable then the 1581 manual. No examples are provided as a tutorial. Instead I

© Copyright 1988 D. W. Martin All Rights Reserved.

suggest that you study the demo programs and source code for the burst routines that are provided on the 1581 Demo/Utilities disk.

BURST READ

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D)
 +CHR$(E);
```

BYTE	BIT	DESCRIPTION
A	7	LOGICAL FLAG (1 = DO LOGICAL TO PHYSICAL TRANSLATION)
	6	IGNORE ERRORS (1 = IGNORE)
	5	NOT USED
	4	SIDE SELECT (SIDE 0 OR 1)
	3-1	SET TO ZERO
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)
B	0-7	DESTINATION TRACK
C	0-7	DESTINATION SECTOR
D	0-7	NUMBER OF SECTORS
E	0-7	NEXT TRACK (OPTIONAL)

PROTOCOL USED: Burst mode handshake.

OUTPUT RESULT: One burst status byte is sent, followed by the burst data. This occurs for each sector transferred. Errors prevent data from being sent unless bit 6 is set to one in byte A.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

BURST WRITE

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D)
 +CHR$(E);
```

BYTE	BIT	DESCRIPTION
A	7	LOGICAL FLAG (1 = DO LOGICAL TO PHYSICAL TRANSLATION)
	6	IGNORE ERRORS (1 = IGNORE)
	5	NOT USED
	4	SIDE SELECT (SIDE 0 OR 1)
	3-2	SET TO ZERO
	1	SET TO ONE
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)
B	0-7	DESTINATION TRACK
C	0-7	DESTINATION SECTOR
D	0-7	NUMBER OF SECTORS
E	0-7	NEXT TRACK (OPTIONAL)

PROTOCOL USED: Burst data is sent to the drive. The host must then do the following for fast serial input. First, pull the clock low and wait for the burst status byte. Then pull the clock high and continue output for multiple sector transfers.

INPUT: The host must transfer burst data.

OUTPUT RESULT: One burst status byte follows each write operation that is performed.

The following hold true for both burst read and write:

RANGE: All track and sector ranges are determined by the current disk format and format translation tables.

CONVENTIONS: Before you can read or write to a diskette, you must first log the disk in using either the **INQUIRE DISK** or **QUERY DISK FORMAT COMMANDS.** You

© Copyright 1988 D. W. Martin All Rights Reserved.

must do this once every time you change a disk also.

* * *

INQUIRE DISK

Command format:

```
PRINT#15,"U0"+CHR$(A);
```

BYTE	BIT	DESCRIPTION
A	7-5	NOT USED
	4	SIDE SELECT (SIDE 0 OR 1)
	3	SET TO ZERO
	2	SET TO ONE
	1	SET TO ZERO
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)

PROTOCOL USED: Burst mode handshake.

OUTPUT RESULT: One burst status byte is sent after each **INQUIRE DISK** operation is performed.

* * *

FORMAT

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D)
 +CHR$(E)+CHR$(F)+CHR$(G);
```

BYTE	BIT	DESCRIPTION
A	7	FORMAT MODE (1 = ACCEPT BYTES THREE TO EIGHT FOR CUSTOM FORMAT. 0 = DO STANDARD 1581 FORMAT MAKE A BAM AND DISK DIRECTORY. DISK NAME = "COPYRIGHT CBM" DISK ID = "86")
	6-4	NOT USED
	3	SET TO ZERO
	2-1	SET TO ONE

© Copyright 1988 D. W. Martin All Rights Reserved.

BYTE	BIT	DESCRIPTION
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)
B	0-7	SECTOR SIZE. THE DEFAULT IS 2 FOR 512 BYTE SECTORS. 1 = 256 BYTE SECTORS 2 = 512 BYTE SECTORS 3 = 1024 BYTE SECTORS
C	0-7	LAST DISK TRACK NUMBER. THE DEFAULT IS 79.
D	0-7	NUMBER OF SECTORS ON A TRACK. DEPENDS ON THE VALUE IN BYTE B. 16 SECTORS PER TRACK FOR 256 BYTE SECTORS. 10 SECTORS PER TRACK FOR 512 BYTE SECTORS. 5 SECTORS PER TRACK FOR 1024 BYTE SECTORS
E	0-7	STARTING TRACK NUMBER. THE DEFAULT IS ZERO.
F	0-7	FILLER BYTE. THE DEFAULT IS $E5/229.
G	0-7	STARTING SECTOR NUMBER. THE DEFAULT IS ONE.

PROTOCOL USED: Uses conventional DOS protocol via the command channel (#15).

OUTPUT RESULT: No output of status. The status is updated within the disk drive.

CONVENTIONS: You must log the disk in using either the **INQUIRE DISK** or **QUERY DISK FORMAT COMMANDS** after using the **FORMAT** command.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

QUERY DISK FORMAT

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B);
```

BYTE	BIT	DESCRIPTION
A	7	FORCE FLAG (1 = STEPS THE HEAD WITH THE OFFSET SPECIFIED IN BYTE B.
	6	NOT USED
	5	SECTOR TABLE (1 = SEND SECTOR TABLE)
	4	SIDE SELECT (SIDE 0 OR 1)
	3	SET TO ONE
	2	SET TO ZERO
	1	SET TO ONE
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)
B	0-7	OFFSET VALUE

PROTOCOL USED: Burst mode handshake.

OUTPUT RESULT: Depends on the following:

Status from track offset of zero: only the burst status byte is sent (no bytes will follow if an error occurred).

Status from track offset set by user:

- burst status byte (no other bytes will follow if an error occurred in compiling the MFM format information)
- number of sectors on a particular track
- logical track number found in the disk header
- the logical sector with the lowest value
- the logical sector with the highest value
- sector interleave (always returns a one)

© Copyright 1988 D. W. Martin All Rights Reserved.

- sector table (sector table is sent when bit 5 in byte A is set

CONVENTIONS: This routine determines the diskette format on any particular track. It will also log in non-standard diskette formats.

* * *

INQUIRE STATUS

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D);
```

BYTE	BIT	DESCRIPTION
A	7	WRITE SWITCH (0 = WRITE)
	6	CHANGE DISK/CHECK FOR CHANGED DISK IF BIT 7 = 0 AND BIT 6 = 1 THEN LOG IN DISK. IF BIT 7 = 1 AND BIT 6 = 1 THEN RETURN WHETHER OR NOT THE DISK WAS LOGGED IN (I.E., $0B ERROR OR PREVIOUS STATUS)
	5	CONTROLS THE SETTING OF THE WRITE AND/OR MASK. BYTE C CONTAINS THE NEW OR MASK, WHILE BYTE D CONTAINS THE AND MASK (WHEN BIT 5 = 1)
	4	SET TO ZERO
	3-2	SET TO ONE
	1	SET TO ZERO
	0	DRIVE NUMBER (ALWAYS 0 ON 1581)
B	0-7	NEW STATUS WHEN BIT 7 ON BYTE A IS ONE
C	0-7	NEW ORA MASK WHEN BIT 5 ON BYTE A IS ONE
D	0-7	NEW AND MASK WHEN BIT 5 ON BYTE A IS ONE

© Copyright 1988 D. W. Martin All Rights Reserved.

PROTOCOL USED: Uses conventional DOS protocol via the command channel (#15) when bit 7 on byte A is zero. If bit 7 on byte A is one then burst mode handshake is used.

OUTPUT RESULT: No output of status when bit 7 on byte A is zero. The burst status byte is sent when bit 7 on byte A is one.

CONVENTIONS: This is the method used to read or write the current status. It may also be used to change the status mask value.

* * *

DUMP TRACK CACHE BUFFER

Command format:

```
PRINT#15,"U0"+CHR$(A)+CHR$(B);
```

BYTE	BIT	DESCRIPTION
A	7	1 = WRITE TRACK CACHE BUFFER TO DISK EVEN IF IT'S NOT "DIRTY".
	6	SIDE SELECT (SIDE 0 OR 1)
	5	NOT USED
	4-2	SET TO ONE
	1	SET TO ZERO
	0	SET TO ONE
B	0-7	PHYSICAL DISK TRACK NUMBER

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

FASTLOAD UTILITY

Command format:

```
PRINT#15,"U0"+CHR$(A)+FILENAME$;
```

BYTE	BIT	DESCRIPTION
A	7	FILE CONTROL BIT. IF SET TO ONE FILE DOES NOT NEED TO BE A PROGRAM FILE. SETTING BIT 7 TO ONE CAN ALLOW YOU TO FASTLOAD PRG, SEQ, AND USR FILES.
	6-5	NOT USED
	4-0	SET TO ONE

The character variable FILENAME$ above contains the name of the file you wish to fast load. It can be from one to 16 characters in length.

PROTOCOL USED: Burst mode handshake.

OUTPUT RESULT: The burst status byte is sent after each sector that is transferred.

The status byte for the **FASTLOAD UTILITY** is decoded as follows:

STATUS BYTE VALUE	MESSAGE
0 OR 1	OK
3 TO 15	FILE READ ERROR
16	FILE NOT FOUND ERROR
31	EOI

NOTE: A byte is sent following the EOI byte that represents the number of data bytes to follow.

* * *

BURST STATUS BYTE DECODED

The burst status byte contains a lot of information for such a small item. It contains information which includes the following: format mode, sector size and controller error messages.

The status byte can be broken down as follows:

© Copyright 1988 D. W. Martin All Rights Reserved.

BIT	DESCRIPTION
7	DISK FORMAT MODE (1 = ALIEN DISK FORMAT 0 = RESIDENT/STANDARD DISK FORMAT)
6	DRIVE NUMBER (ALWAYS 0 ON 1581)
5-4	DISK SECTOR SIZE
	00 = 128 BYTE SECTORS (WHICH IS NOT SUPPORTED ON THE 1581)
	01 = 256 BYTE SECTORS
	10 = 512 BYTE SECTORS (WHICH IS THE DEFAULT VALUE ON THE 1581)
	11 = 1024 BYTE SECTORS
3-0	DISK CONTROLLER STATUS

Disk Controller Status

0000 = OK
0001 = OK
0010 = CANNOT FIND HEADER BLOCK
0011 = NO ADDRESS MARK FOUND
0100 = DATA BLOCK NOT PRESENT
0101 = DATA CRC ERROR
0110 = DISK FORMAT ERROR
0111 = DISK VERIFY ERROR
1000 = DISK WRITE PROTECT ERROR
1001 = HEADER BLOCK CRC ERROR
1010 = DISK IS WRITE PROTECTED
1011 = DISK CHANGE OCCURRED
1100 = DISK FORMAT IS ILLOGICAL
1101 = RESERVED
1110 = SYNTAX ERROR OCCURRED
1111 = NO DISK DRIVE PRESENT

* * *

BURST ROUTINE NOTES

The most important thing to mention about the burst commands is that most of them must be executed by machine language programs since BASIC is not quite up to handling the burst protocols speed.

The burst command must be executed by a machine language command whenever it specifies the use of the burst mode handshake. Other commands can be executed by BASIC programs (i.e. the burst format command).

© Copyright 1988 D. W. Martin All Rights Reserved.

Therefore, you must convert the BASIC command format in the text above to it's machine language equivalent.

Using BASIC to show the command format was chosen because it is the language that most users are familiar with and the easiest to understand. Experienced programmers will find it easy to convert the BASIC statements to their machine language equivalents.

If you not interested in programming and only want to use the burst routines in your own programs then I suggest that you check out the sample burst routine files on your 1581 Demo/Utilities disk.

Another thing I might mention is that all burst commands use track and sector parameters that refer to physical disk locations. The only exceptions to this rule are the burst READ and WRITE commands. These commands allow you to chose between logical or physical disk translation.

The burst protocol handshake is outlined on page 99 of the 1581 User's Guide. Study it and the machine language source files on the demo disk. They will help you to understand the handshaking being used on the fast serial bus. Also, it might not be a bad idea if you read chapters nine and ten in the 1581 User's Guide.

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 4 - Burst Utility Commands

The following commands are new additions to the Commodore DOS world. They are all executed via the command channel (#15) and are listed and explained below. These commands are explained using my interpretation of what they do. Unfortunately, Commodore did not document these commands in the 1581 manual. This was a glaring omission in an otherwise perfect drive manual.

Serial Bus Mode

The serial bus mode command tells the 1581 to use fast or slow serial transfers to and from the computer.

The command takes the form:

```
PRINT#15,"U0>Bx"
```

where "x" equals one for fast serial and zero for slow serial. Apparently this command was added to turn burst mode on and off when using the C128 with the 1581. It does not seem to effect C64 usage.

DOS Sector Interleave

This command controls the number of sectors between each linked block on a disk. The default value is one on power up. This value makes the drive very fast when accessing disk data.

Increasing it's value would make the drive run a little slower since it would have to search a little further for data from block to block.

The command takes the form:

```
PRINT#15,"U0>Sx"
```

where "x" equals a value from 0 to 255. I see no real reason for having this command and recommend that you leave the interleave set to one on the 1581.

For comparison the 1541 and 1571 (C128 mode) have normal sector interleave rates of ten and six respectively. You may also find that some folks use the term "skew rate" or just "skew. " These terms of synonymous with sector interleave.

© Copyright 1988 D. W. Martin All Rights Reserved.

DOS Retries

This command allows you to select the number of read attempts that the DOS tries on bad sectors. The default value is two on power up.

The command takes the form:

```
PRINT#15,"U0>Rx"
```

where "x" is a value from 0 to 255. A value of zero tells the drive to not try and read the bad sector again, but to just except it as a bad one. When "x" has a value from 1 to 255 the DOS will try to reread a sector "x" times. Once again I recommend that you leave this value at it's default of two.

One note of interest about this command is that in the 1541 the number of retries could be changed to prevent head banging on 1541 protected diskettes. That opens the possibilities of using the same trick on the 1581, but I have not found reason to do that yet since there are no copy protected 1581 disks out there.

Also increasing the value will often allow you to recover data from a damaged sector. But, beware, over working the drive might wear the drive mechanism.

ROM Signature Analysis

The 1581 DOS provides a means of extensively testing the ROMs in the drive. Simply issue the following command:

```
PRINT#15,"U0>T"
```

The drive will now test your 1581's ROM. If a failure in the ROM is detected the LED's will blink continuously in a four blink pattern.

Verify Select

This command is used to turn the 1581's verify mode on and off. During writing this command will control whether or not the drive verify's the data that is written to the diskette.

The command takes the form:

```
PRINT#15,"U0>Vx"
```

© Copyright 1988 D. W. Martin All Rights Reserved.

where "x" equals one for verify off and zero for verify on.

I recommend the usage of this command when using the disk drive for important work. It's a fact that using verify mode certainly cannot hurt anything. However, it might slow the drive up a little since verifying data takes a little extra time.

One important thing to remember is DON'T PANIC! Your 1581 will prove to be a very reliable device when reading and writing to diskettes. I find however that repeated writing and reading to a diskette seems to invite errors more and more often. I suggest that people using the 1581 on BBSes or as a work drive always turn verify mode on.

Disk Device Select

Although the 1581 provides easy access to changing the device number settings (via DIP switches), it can often easier to change the device number through software. Commodore made software device changes possible with the following command:

```
PRINT#15,"U0>"+CHR$(x)
```

where "x" represents a device number from four to thirty. Standard disk drive devices are numbered eight through eleven.

Any device number changes made with this command are not permanent or used as the default. If you want a device number to be the default the best way to accomplish this is to read the section in your 1581 manual on hardware device setting. This appears on page 111 of the 1581 User's Guide.

Mystery Command - SIEEE Timing

The 1581 also provides an undocumented utility command for which I could find only one use. This command takes the form of:

```
PRINT#15,"U0>Ix"
```

where "x" can be any value from zero to 255.

This command controls the disk drive's SIEEE timing value. The SIEEE timing value is stored at $9D/157. On reset this value is set to $20/32 by a routine at $B091. This command may have had something to do with a now nonexistant IEEE interface for the

© Copyright 1988 D. W. Martin All Rights Reserved.

1581.

Burst Memory Read/Write

Burst memory read and write use the standard burst protocol with no status byte. They take the form of:

```
PRINT#15,"U0>MR"+CHR$(>MEMADDR)+CHR$(# OF PAGES)

                        OR

PRINT#15,"U0>MW"+CHR$(>MEMADDR)+CHR$(# OF PAGES)
```

These commands are for C128 users only and files showing how they work can be found on the 1581 Demo/Utilities disk. As well as files allowing you to access the 1581's burst protocol features.

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 5 - Using Kracker-Mon

Exploring the 1581, a New Frontier

Now you to can be a brave 1581 explorer and discover new worlds within your 1581 disk drive. A vast frontier of new DOS routines and functions are now available for exploration and discovery in the 1581 disk drive. The utility described in this chapter will be your vehicle to discovery.

Kracker-Mon

Kracker-Mon is an all purpose machine language monitor (MLM) with many features including but not limited to:

Kracker-Mon Relocator
Op-Code Editor
Computer MLM
Drive-Mon
DOS Wedge

The program and documentation were provided courtesy of KRACKER JAX, a division of Software Support International. Kracker-Mon was written by Mike Howard. Thanks guys for this wonderful tool!

DOS in the Dumps

For your convenience I have provided a program in Appendix A that will dump a copy of the 1581's 32K ROM operation system onto a diskette so that you can explore the DOS with other tools such as disassemblers, etc.

Simply key in the program called DOS DUMP, save it to a blank disk in your 1581 disk drive. Proofread it before execution and if everything's ok run it.

After running DOS DUMP it will prompt you for the source drive (where it reads the ROM) and the destination disk (where it wants to put the DOS file). This program is not extremely fast so go read a book or whatever for a few minutes.

The program is compatible with the 1571 also and will dump a copy of the DOS on a C64 or C128. It does not use burst routines, since they are not C64 compatible. If you choose to you can use Kracker-Mon to dump the DOS, but doing 32K of ROM that way is slow and tedious. Sit back and let the computer do it using DOS DUMP. Put that computer to work folks!

© Copyright 1988 D. W. Martin All Rights Reserved.

How do you do that Kracker-Mon?

The following documentation for Kracker-Mon covers program operation only. If you have purchased this book with it's companion disk please refer to Appendix A for information on how to load Kracker-Mon.

Kracker-Mon is completely relocatable in memory. The + and - keys will increment and decrement the monitor address. Hitting the RETURN key while the "Monitor=$X000" is highlighted will also increment the monitor.

F1 - Directory of disk in drive.

F3 - Exit Kracker-Mon, enter BASIC

RESTORE - Restart program to beginning menu

Option 1 - Execute the chosen monitor

Option 2 - Save chosen monitor to a WORK DISK

Option 3 - Edit the opcode file called OPS on any WORK DISK.

Notes on Option 2

Selecting option 2 saves the following information onto a WORK DISK:

```
Autoboot filename - "MONX000"
Opcode listings   - "OPS"
Monitor           - "X0"
```

Execute a saved monitor via:

```
LOAD "MONX000",8,1
```

It will now load and execute itself.

Notes on Option 3

CURSOR U/D - Slow scroll through the list

CURSOR R/L - Fast scroll through the list

RESTORE - Reset to previous menu

SPACE BAR - Allows you to change the mnemonic

© Copyright 1988 D. W. Martin All Rights Reserved.

A - Steps through the addressing modes (changes them)

HOME - Returns cursor to beginning ($00 byte)

S - Saves changed opcode file to WORK DISK

Note: Old opcode file is overwritten so be sure that you do not overwrite an OPS file that you need my mistake. Rename the old file or use another WORK DISK. **NEVER save anything to the companion diskette!**

Kracker-Mon Commands

Command	Description
R	Displays status of A, X, and Y registers stack pointer and program counter.
G xxxx	Executes code starting at $xxxx
X	Returns user to BASIC.
M FFFF LLLL	Displays in hex and ASCII memory between the two addresses. If the second address is not specified, scrolls forever. RUN/STOP halts.
@	Send a DOS command. I used alone returns the drive status. @$ gives a disk directory. Space pauses the directory and stop aborts.
L	Load file from disk into memory.
L "FILENAME",device#,address(optional)	If an address is given the program loads at that memory address.
V	Verify file in memory with one on disk. Same format as load command. "?" stands for verify error.

© Copyright 1988 D. W. Martin All Rights Reserved.

V "FILENAME",device#,address(optional)

S Save file in memory to disk.

S "FILENAME",device#,FFFF,LLLL+1

Example: S "FILE",08,C000,D000
saves memory from $C000-$CFFF

F FFFF LLLL XX Fills memory from $FFFF to $LLLL with $XX byte value.

D FFFF LLLL Disassembles memory. You can use the cursor U/D keys to scroll through the disassembly. If the LLLL parameter is not specified one page of disassembly is printed instead of a range of memory.

P Send code to printer.

PD FFFF LLLL sends disassembly to the printer.

PM FFFF LLLL sends hex memory listing to the printer.

(1525 compatible printers only!)

A xxxx Assemble mnemonic commands. Assembles code beginning at $xxxx (Be sure to use proper spacing between characters.)

H FFFF LLLL PP Hunts from $FFFF to $LLLL for up to an eight byte pattern (PP). Use single quotes around ASCII test to hunt for ASCII text. ASCII and HEX may be mixed.

Examples:

H E000 FFFF 'COMMODORE'

H E000 FFFF 43 4F 4D 4D

© Copyright 1988 D. W. Martin All Rights Reserved.

H E000 FFFF 43 4F 'MM'

T — Transfers memory from $FFFF to $LLLL to $xxxx.

Examples:

T FFFF LLLL xxxx

T E000 FFFF 1000

TC — Same syntax as T command. Will transfer computer memory to drive.

TD — Same syntax as T command. Will transfer drive memory to the computer.

TF — Same syntax as T command. Fast command version of TC. Warning! $xxxx cannot be between $0001 and $0147. **NOT 1581 COMPATIBLE!**

O — This is the letter O, not zero. O followed by an 8, 9, A, or B (device number) will out you in the Drive-Mon mode for the specified drive. The above commands are the same for the Drive-Mon except the P feature is disabled.

For printer listings of drive memory, send the code to the computer, then the printer. O and RETURN sends you back to the computer memory. A "]" lets you know your in drive memory, while a "." denotes computer memory.

To assemble/disassemble beneath the ROMs and the VIC CHIP, change location $0002 as if it were $0001. $0001 cannot be changed through the monitor.

$0002: $37 All ROMs in

$36 Bank out BASIC ($A000-$BFFF)

© Copyright 1988 D. W. Martin All Rights Reserved.

$35 Bank out KERNAL and BASIC ($A000-$BFFF and $E000-$FFFF)

$30 Bank in RAM under $D000

$31 Bank in character ROM under $D000

© Copyright 1988 D. W. Martin All Rights Reserved.

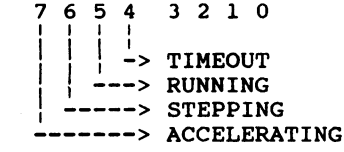
1581 System Overview

$0000-$1FFF	8K of RAM memory
$0000-$00FF	Zero page variables
$0100-$01FF	Vectors, variables, and stack area
$0200-$0BFF	DOS data buffers
$0C00-$1FFF	Track cache buffer
$4000-$400F	Serial bus I/O controller 8520A 2 Mhz CIA chip
$6000-$6003	MFM Floppy disk controller WD177x series chip
$8000-$FFFF	32K of DOS in ROM containing 1581 controller routines

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$0000-$0001	RAMTST	POINTER USED BY RAM AND ROM TESTS
$0002-$000A	JOBS	JOB QUEUE FOR JOBS 0 TO 8
	JOB0	$0002 BUFFER 0 AT $0300
	JOB1	$0003 BUFFER 1 AT $0400
	JOB2	$0004 BUFFER 2 AT $0500
	JOB3	$0005 BUFFER 3 AT $0600
	JOB4	$0006 BUFFER 4 AT $0700
	JOB5	$0007 BUFFER 5 AT $0800
	JOB6	$0008 BUFFER 6 AT $0900
	JOB7	$0009 BUFFER 7 AT $0A00 (BAM 0)
	JOB8	$000A BUFFER 8 AT $0B00 (BAM 1)
$000B-$001C	HDRS	TRACK AND SECTORS FOR JOBS LISTED ABOVE
	TSJ0	$000B, $000C TRACK AND SECTOR JOB 0
	TSJ1	$000D, $000E TRACK AND SECTOR JOB 1
	TSJ2	$000F, $0010 TRACK AND SECTOR JOB 2
	TSJ3	$0011, $0012 TRACK AND SECTOR JOB 3
	TSJ4	$0013, $0014 TRACK AND SECTOR JOB 4
	TSJ5	$0015, $0016 TRACK AND SECTOR JOB 5
	TSJ6	$0017, $0018 TRACK AND SECTOR JOB 6

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
	TSJ7	$0019, $001A TRACK AND SECTOR JOB 7
	TSJ8	$001B, $001C TRACK AND SECTOR JOB 8
$001D-$001E	DSKID	MASTER COPY OF DISK ID
$001F-	HEADER	PHYSICAL TRACK NUMBER
$0021-		PHYSICAL SECTOR NUMBER
$0022-		PHYSICAL SECTOR LENGTH (2 = 512 BYTES FOR THE 1581) 0 = 128 BYTES IN THE SECTOR 1 = 256 BYTES IN THE SECTOR 2 = 512 BYTES IN THE SECTOR (NORMAL) 3 = 1024 BYTES IN THE SECTOR
$0025-	WPSW	WRITE PROTECT CHANGE FLAG
$0026-	DRVST	LCC VARIABLE  $00 NO DRIVE ACTIVE $20 DRIVE IS RUNNING $30 RUNNING AND TIMEOUT $50 STEPPING / RUNNING $80 ACCELERATING
$0027-	DRVTRK	CURRENT PHYSICAL TRACK BEING ACCESSED

© Copyright 1988 D. W. Martin All Rights Reserved.

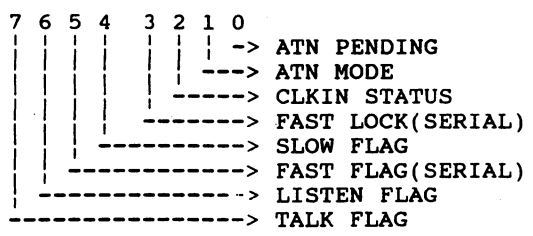
LOCATION	LABEL	DESCRIPTION
$0028-	CMD	TEMP COMMAND JOB
$0029-	CMDSIZ	COMMAND STRING SIZE
$002A-$002B	ACLTIM	ACCELERATION TIME DELAY
$002C-	SAVSP	SAVE STACK POINTER
$002D-	AUTOFG	AUTO INITIALIZE FLAG
$002E-	SECINC	SECTOR INTERLEAVE FOR SEQ (SET ON RESET $01)
$002F-	FREBLK	FREE BLOCKS COUNTER
$0030-	REVCNT	ERROR RECOVERY COUNT (SET ON RESET TO $02)
$0031-$0032	BMPNT	BIT MAP POINTER (SET ON RESET TO $0000)
$0033-$0034	USRJMP	USER JUMP TABLE POINTER (SET ON POWER UP OR RESET TO $FFEA)
$0035-	WBAM	BAM STATUS FLAG (0 = CLEAN)
$0036-$0037	CTMP	TEMPORARY WORK AREA
$0038-$003E	TMP	TEMPORARY WORK AREA
$003F-	TMPJBN	TEMPORARY JOB NUMBER
$0040-	T0	
$0041-	T1	
$0042-	T2	TEMPORARY WORK AREA ($0040-$0044)
$0043-	T3	
$0044-	T4	
$0046-$004B	IP	INDIRECT POINTER VARIABLE (SET TO $0100 ON POWERUP OR RESET)
$004C-	PRGTRK	LAST PROGRAM ACCESSED

© Copyright 1988 D. W. Martin All Rights Reserved.

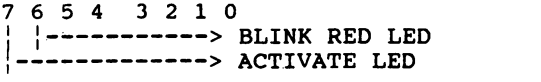
LOCATION	LABEL	DESCRIPTION
$004D-	TRACK	CURRENT TRACK
$004E-	SECTOR	CURRENT SECTOR
$004F-	TOS	TOP OF STACK
$0050-	LINDX	LOGICAL INDEX
$0051-	EOIFLG	TEMP EOI
$0052-	SA	SECONDARY ADDRESS
$0053-	ORGSA	ORIGIANL SA
$0054-	DATA	TEMPORY DATA BYTE
$0055-	R0	
$0056-	R1	
$0057-	R2	TEMPORARY RESULTS ($0055-$0059)
$0058-	R3	
$0059-	R4	
$005A-$005E	RESULT	RESULT AREA
$0060-$0064	ACCUM	ACCUMULATOR
$0065-$0066	DIRBUF	DIRECTORY BUFFER $05/$02
$0067-	F1PTR	FILE STREAM 1 POINTER
$0068-	RECPTR	POINTER TO START OF RECORD
$0069-	SSNUM	NUMBER OF SIDE SECTOR
$006A-	SSIND	INDEX TO SIDE SECTOR
$006B-	RELPTR	RELATIVE FILE POINTER TO TRACK
$006C-	JOBNUM	CURRENT JOB NUMBER
$006D-	BUFUSE	BUFFER ALLOCATION

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$006E-	LSTDRV	LAST DRIVE WITHOUT ERROR. USED AS THE DEFAULT DRIVE NUMBER. ALSO USED AS THE NO DRIVE FLAG.
$006F-	CDOSVER	COPY OF DOS VERSION "D" FOR COMPARISON
$0070-	LINUSE	LINDX USE WORD
$0071-	DIRSEC	DIRECTORY SECTOR
$0072-	DELSEC	SECTOR OF FIRST AVAILABLE ENTRY
$0073-	DELIND	INDEX OF FIRST AVAILABLE ENTRY
$0074-	LBUSED	LAST BUFFER USED
$0075-	NUMSEC	WORK AREA FOR FINDING BEST SECTOR TO DO/LOGICAL NUMBER OF SECTORS
$0076-	FSFLAG	FAST SERIAL FLAG



$0077-	LSNADR	LISTENER ADDRESS ($28 ON RESET IF 1581 IS SET VIA DIP SWITCH TO DEVICE 8)
$0078-	TLKADR	TALKER ADDRESS ($48 ON RESET IF 1581 IS SET VIA DIP SWITCH TO DEVICE 8)
$0079-	LEDPRINT	LED VARIABLE



© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$007A-	TEMPSA	TEMPORARY SECONDARY ADDRESS
$007B-	CMDWAT	COMMAND WAITING FLAG
$007C-	U0CMDNO	"U0" COMMAND NUMBER
$007D-	CTRLST	CONTROLLER STATUS
$007E-$007F	BUFPNT	BUFFER POINTER
$0080-	DKMODE	BURST STATUS
$0081-	JOBRTN	JOB RETURN FLAG
$0082-	YREG	CPU Y REGISTER STORAGE
$0083-	NEXTJOB	CONTROLLER NEXT JOB
$0084-	CINDEX	CONTROLLER JOB INDEX
$0085-$0086	INFO	CONTROLLER JOB INFORMATION
$0087-	TCDIRTY	TRACK CACHE DIRTY FLAG
$0088-	CMDTRK	CONTROLLER DESTINATION TRACK
$0089-	DKANDMASK	BURST STATUS MASK (AND)
$008A-	DKORAMASK	BURST STATUS MASK (ORA)
$008B-$008C	CACHE	POINTER TO BUFFER CACHE ($0C00)
$008D-	IOBYTE	VERIFY / CRC / REL FILE FLAG

```
7 6 5 4  3 2 1 0
| | |------------> REL FILE
| |--------------> CRC CHECK
|----------------> VERIFY CHECK
```

(NOTE: THIS IS BYTE $06/6 IN THE BAM)

$008E-	PSTARTRK	PHYSICAL STARTING TRACK
$008F-	PMAXTRK	PHYSICAL ENDING TRACK
$0090-	STARTRK	LOGICAL STARTING TRACK
$0091-	PSECTRSIZ	PHYSICAL SECTOR SIZE

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$0092-	PNUMSEC	PHYSICAL NUMBER OF SECTORS/SIDE
$0093-	PENDSEC	PHYSICAL ENDING SECTOR
$0094-	PSTARTSEC	PHYSICAL STARTING SECTOR
$0095-	CACHETRK	CURRENT PHYSICAL TRACK CACHE
$0096-	TCACHESID	TRANSLATED CURRENT TRACK CACHE SIDE
$0097-	CACHESIDE	CURRENT TRACK CACHE SIDE
$0098-	SETVAL	SETTLING TIME VALUE
$0099-	HDRJOB	SHIFTED NEXT JOB / HEADER POINTER
$009A-	GAP3	FORMAT GAP
$009B-	FILLBYTE	FORMAT FILL BYTE
$009C-	SIEEETIM	SIEEE TIMING COUNTER
$009D-	SIEEESET	SIEEE TIMING VALUE
$009E-	BLINK	ERROR BLINKING COUNTER
$009F-$00A7	CACHEOFF	OFFSET INTO THE TRACK CACHE BUFFER (1 BYTE PER JOB)
$00A8-$00BA	LINTAB	SA:LINDX TABLE THIS TABLE INDICATES THE CURRENT STATUS OF EACH DATA CHANNEL (SECONDARY ADDRESS) EACH POSITION REPRESENTS ONE CHANNEL: CHANNEL 0 = $00A8 CHANNEL 1 = $00A9 ETC.

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
		POSSIBLE CHANNEL STATUS VALUES ARE: $FF - INACTIVE $81 - OPEN FOR WRITE $41 - READ/WRITE $01 - OPEN FOR READ
$00BB-$00D0	BUFTAB	BUFFER BYTE POINTERS THESE POINTERS FOR EACH BUFFER ARE USED TO POINT AT THE NEXT BYTE IN THE BUFFER TO BE USED. THE "B-P" COMMAND SETS THESE POINTERS. $00BB-$00BC BUFFER #0($0300) $00BD-$00BE BUFFER #1($0400) $00BF-$00C0 BUFFER #2($0500) $00C1-$00C2 BUFFER #3($0600) $00C3-$00C4 BUFFER #4($0700) $00C5-$00C6 BUFFER #5($0800) $00C7-$00C8 BUFFER #6($0900) $00C9-$00CA BUFFER #7($0A00) BAM 0 $00CB-$00CC BUFFER #8($0B00) BAM 1 $00CD-$00CE CMD BUFFER ($0200) $00CF-$00D0 ERR BUFFER ($02D0)
$00D1-$00D7	BUF0	TABLE OF CHANNEL NUMBERS ASSIGNED TO EACH OF THE BUFFERS. $FF = INACTIVE BUFFER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$00D8-$00DE	BUF1	TABLE OF CHANNEL NUMBERS ASSIGNED TO EACH OF THE BUFFERS. $FF = INACTIVE BUFFER
$00DF-$00E3	LRUTBL	LEAST RECENTLY USED TABLE
$00E4-	RELPTR	RELATIVE FILE POINTER TO TRACK
$00E5-$00E9	ENTSEC	SECTOR OF DIRECTORY ENTRIES
$00EA-$00EE	ENTIND	INDEX OF DIRECTORY ENTRIES
$00EF-$00F3	FILDRV	DEFAULT FLAG, DRIVE NUMBER (ALL 0 ON 1581)
$00F4-$00F8	PATTYP	PATTERN, REPLACE, CLOSED FLAGS, TYPE
$00F9-$00FE	FILTYP	CHANNEL FILE TYPE
$0100-	STACK	START OF DRIVE STACK AREA
$0101-	RELVAR	VARIABLE FOR RELATIVE FILES ($0100-$0110)
$0102-	MARKSIDE	MARK SIDE SECTOR AS
$0104-	BRKJMP	JUMP TO CONTROLLER ROUTINE VIA "BRK"
$0109-	SIDTRK	SIDE SECTOR TRACK INFORMATION
$0110-	SIDSEC	SIDE SECTOR SECTOR INFORMATION
$0190-$01BB	SVECTS	JUMP TABLE OF MAJOR DOS ROUTINES EXECUTE VIA JSR ($XXXX)
$0190-$0191	VIDLE	INDIRECT FOR IDLE
$0192-$0193	VIRQ	INDIRECT FOR IRQ
$0194-$0195	VNMI	INDIRECT FOR NMI
$0196-$0197	VVERDIR	VALIDATE COMMAND

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$0198-$0199	VINTDRV	INITIALIZE COMMAND
$019A-$019B	VPART	PARTITION COMMAND
$019C-$019D	VMEM	MEMORY READ/WRITE COMMANDS
$019E-$019F	VBLOCK	BLOCK COMMANDS
$01A0-$01A1	VUSER	USER COMMANDS
$01A2-$01A3	VRECORD	RECORD COMMAND FOR REL FILES
$01A4-$01A5	VUTLODR	UTILITY LOADER COMMAND
$01A6-$01A7	VDSKCPY	COPY COMMAND
$01A8-$01A9	VRENAME	RENAME COMMAND
$01AA-$01AB	VSCRTCH	SCRATCH COMMAND
$01AC-$01AD	VNEW	NEW COMMAND (FORMAT DISK)
$01AE-$01AF	VERROR	CONTROLLER ERROR HANDLER
$01B0-$01B1	VATNSRV	ATN SERVER FOR SERIAL BUS
$01B2-$01B3	VTALK	TALK ROUTINE FOR SERIAL BUS
$01B4-$01B5	VLISTEN	LISTEN ROUTINE FOR SERIAL BUS
$01B6-$01B7	VLCC	DISK CONTROLLER ROUTINE
$01B8-$01B9	VTRANS_TS	TRANSLATES LOGICAL SECTORS TO PHYSICAL SECTORS
$01BA-$01BB	VCMDERR	DOS ERROR HANDLER
$01BC-$01CD	HDRS2	CONTAINS THE TRANSLATED PHYSICAL TRACK AND SECTOR FOR EACH OF THE JOBS IN THE JOB QUEUE (JOBS 0 TO 8, 2 BYTES FOR EACH JOB)
$01CE-$01D6	SIDS	CONTAINS THE PHYSICAL SIDE FOR EACH OF THE JOBS IN THE JOB QUEUE (JOBS 0 TO 8, 1 BYTE FOR EACH JOB)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$01D7-	CTLTIMH	CONTROLLER TIMER VARIABLE HIGH
$01D8-	CTLTIML	CONTROLLER TIMER VARIABLE LOW
$01D9-	MOTORACC	ACCELERATION STARTUP
$01DA-$01E4	WDCMDS	WD177X CONTROLLER COMMANDS (SEE CHAPTER 7)
$01DA-	WDRESTORE	$08
$01DB-	WDSEEK	$18
$01DC-	WDSTEP	$28
$01DD-	WDSTEPIN	$48
$01DE-	WDSTEPOUT	$68
$01DF-	WDREADSECTOR	$88
$01E0-	WDWRITESECTOR	$AA
$01E1-	WDREADADDRESS	$C8
$01E2-	WDREADTRACK	$E8
$01E3-	WDWRITETRACK	$FA
$01E4-	WDFORCEIRQ	$D0
$01E5-	DIRST	STARTING DIRECTORY SECTOR
$01E6-$01E9	SAVECTS	SAVE AREA FOR VECTORS
$01EA-	BURSTST	BURST CONTROLLER STATUS
$01EB-	VERNUM	DOS VERSION NUMBER
$01EC-	DOSTYP	DOS TYPE
$01ED-	PARTHI	HIGH PARTITION COUNT
$01EE-	PARTLO	LOW PARTITION COUNT
$01EF-	MINSEK	BURST MINIMUM SECTOR NUMBER
$01F0-	MAXSEK	BURST MAXIMUM SECTOR NUMBER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$01F1-$01F9	BUFIND	BUFFER INDIRECTS
$01FA-	WPSTAT	WRITE PROTECT STATUS $00 - NOT PROTECTED $08 - PROTECTED
$01FB-	DEJAVU	AUTOBOOT FLAGS 7 6 5 4 3 2 1 0 | | | |---> BOOT ON INITDRV |-----> BOOT ON RESET
$01FC-$01FE	FLOAD	EXECUTE BURST FAST LOAD VIA JSR $01FC OR JMP $01FC CONTAINS: JMP $BA40
$0200-$0229	CMDBUF	COMMAND BUFFER
$022A-	CMDNUM	COMMAND NUMBER
$022B-	DIRTRK	SYSTEM TRACK ($28)
$022C-	MAXTRK	MAXIMUM TRACK + 1 ($51)
$022D-	TYPE	ACTIVE FILE TYPE
$022E-	F1CNT	FILE STREAM 1 COUNT
$022F-	F2CNT	FILE STREAM 2 COUNT
$0230-	F2PTR	FILE STREAM 2 POINTER
$0231-	FILCNT	COUNTER, FILE ENTRIES
$0232-	INDEX	CURRENT INDEX IN BUFFER
$0233-	TYPFLG	MATCH BY TYPE FLAG
$0234-$023A	CHNRDY	CHANNEL STATUS
$023B-$0241	CHNDAT	CHANNEL DATA BYTE
$0242-$0248	LSTCHR	CHANNEL LAST CHARACTER POINTER
$0249-$024F	RECL	TABLE OF LOW BYTES OF RECORD NUMBERS FOR EACH BUFFER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$0250-$0256	RECH	TABLE OF HIGH BYTES OF RECORD NUMBERS FOR EACH BUFFER
$0257-$025D	NR	TABLE OF NEXT RECORD NUMBER FOR BUFFERS
$025E-$0264	RS	TABLE OF RECORD SIZE FOR EACH BUFFER
$0265-$026B	SS	TABLE OF SIDE SECTORS FOR EACH BUFFER
$026C-	STRSIZ	LENGTH OF THE STRING
$026D-	ENTFND	DIRECTORY ENTRY FOUND FLAG
$026E-	DIRLST	DIRECTORY LISTING FLAG
$026F-	REC	RECORD SIZE USED BY DIRECTORY ROUTINES
$0270-	TRKSS	SIDE SECTOR TRACK USED BY DIRECTORY ROUTINES
$0271-	SECSS	SIDE SECTOR SECTOR USED BY DIRECTORY ROUTINES
$0272-$027C	LSTJOB	LAST JOB BY BUFFER
$027D-$0283	DSEC	SECTOR OF DIRECTORY ENTRY BY BUFFER
$0284-$028A	DIND	INDEX OF DIRECTORY ENTRY BY BUFFER
$028B-	PRGSEC	LAST PROGRAM SECTOR
$028C-	WLINDX	WRITE LINDX
$028D-$028E	NBTEMP	NUMBER OF BLOCKS TEMPORARY
$028F-	CHAR	CHARACTER UNDER THE PARSER
$0290-	LIMIT	PTR LIMIT IN COMPARISON
$0291-$0296	FILTBL	TABLE OF FILENAME POINTERS
$0297-$029B	FILTRK	FIRST FILE LINK (TRACK)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$029C-$02A0	FILSEC	FIRST FILE LINK (SECTOR)
$02A1-	PATFLG	PATTERN PRESENT FLAG
$02A2-	IMAGE	FILE STREAM IMAGE
$02A3-	DRVCNT	NUMBER OF DRIVE SEARCHES
$02A4-	DRVFLG	DRIVE SEARCH FLAG
$02A5-	FOUND	FOUND FLAG IN DIRECTORY SEARCHES
$02A6-	LSTBUF	EQUALS 0 IF LAST BLOCK
$02A7-	MODE	ACTIVE FILE MODE (R, W)
$02A8-	JOBRTN	JOB RETURN FLAG

```
7 6 5 4   3 2 1 0
| |
| |----> SKIP TRK & SEC CHECK
|------> RETURN ON ERROR
```

$02A9-	NDBL	NUMBER OF DISK BLOCKS FREE LOW
$02AA-	NDBH	NUMBER OF DISK BLOCKS FREE HIGH
$02AB-	ERWORD	ERROR WORD ON RECOVERY
$02AC-$02CF	NAMBUF	DIRECTORY BUFFER
$02D0-$02F4	ERRBUF	ERROR MESSAGE BUFFER
$0300-$03FF	BUFF0	DATA BUFFER #0
$0400-$04FF	BUFF1	DATA BUFFER #1
$0500-$05FF	BUFF2	DATA BUFFER #2
$0600-$06FF	BUFF3	DATA BUFFER #3
$0700-$07FF	BUFF4	DATA BUFFER #4
$0800-$08FF	BUFF5	DATA BUFFER #5
$0900-$09FF	BUFF6	DATA BUFFER #6

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$0A00-$0AFF	BAM1	BAM FOR TRACKS 1 TO 40
$0B00-$0BFF	BAM2	BAM FOR TRACKS 41 TO 80
$0C00-$1FFF	BUFCACHE	TRACK CACHE BUFFER (20 PAGES)

© Copyright 1988 D. W. Martin All Rights Reserved.

Serial I/O 8520A ($4000-$400F)

LOCATION	BIT	DESCRIPTION
$4000	0-7	DATA PORT A (FLOPPY DISK CONTROLLER I/O)
DSKCH	7	DISK FLAG (1 = PRESENT)
ACTLED	6	GREEN LED (1 = ON)
ERRLED	5	RED LED (1 = BRIGHT)
DEVSW1	4	DEVICE SWITCH 1 (LEFT)
DEVSW2	3	DEVICE SWITCH 2 (RIGHT)
MOTOR	2	0 = MOTOR ON/1 = MOTOR OFF
DRVRDY	1	DRIVE READY (1 = DRV READY)
SIDE	0	0 = SIDE 0/1 = SIDE 1
	* * *	
$4001	0-7	DATA PORT B (SERIAL DATA I/O)
ATNIN	7	ATTENTION IN LINE
WRTPRO	6	WRITE PROTECT (1 = WRITE ENABLE/0 = WRITE PROTECTED)
BUSDIR	5	1581 BUS DATA DIRECTION (0 = INPUT/1 = OUTPUT)
ATNACK	4	ATTENTION ACKNOWLEDGE LINE
CLKOUT	3	CLOCK OUT LINE
CLKIN	2	CLOCK IN LINE
DATOUT	1	DATA OUT LINE
DATAIN	0	DATA IN LINE
$4002	DDA 0-7	DATA DIRECTION PORT A

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION		BIT	DESCRIPTION
$4003	DDB	0-7	DATA DIRECTION PORT B
$4004	TAL	0-7	TIMER A LOW/BAUD RATE LOW
$4005	TAH	0-7	TIMER A HIGH/BAUD RATE HIGH
$4006	TBL	0-7	TIMER B LOW/CTRLER IRQ
$4007	TBH	0-7	TIMER B HIGH/CTRLER IRQ
$4008	SEC	0-7	USED FOR DISK CHANGE DETECTOR
$400C	SDR	0-7	SERIAL DATA REGISTER
$400D	ICR	0-7	INTERRUPT CONTROL REGISTER
$400E	CRA	0-7	CONTROL REGISTER A
$400F	CRB	0-7	CONTROL REGISTER B

* * *

WD177X Floppy Disk Controller

LOCATION	READ FUNCTION	WRITE FUNCTION
$6000	STATUS	COMMAND
$6001	TRACK	TRACK
$6002	SECTOR	SECTOR
$6003	DATA	DATA

* * *

WD177X Command Summary

TYPE	DESCRIPTION	COMMAND VALUE BIT 7 6 5 4 3 2 1 0
I	RESTORE	0 0 0 0 H V X Y
I	SEEK	0 0 0 1 H V X Y
I	STEP	0 0 1 U H V X Y

© Copyright 1988 D. W. Martin All Rights Reserved.

TYPE	DESCRIPTION	COMMAND VALUE BIT 7 6 5 4 3 2 1 0
I	STEP IN	0 1 0 U H V X Y
I	STEP OUT	0 1 1 U H V X Y
II	READ SECTOR	1 0 0 M H E 0 0
II	WRITE SECTOR	1 0 1 M H E P A
III	READ ADDRESS	1 1 0 0 H E 0 0
III	READ TRACK	1 1 1 0 H E 0 0
III	WRITE TRACK	1 1 1 1 H E P 0
IV	FORCE INTERRUPT	1 1 0 1 I J K L

BIT KEY

H: 0 = TURN MOTOR ON, 1 = TURN MOTOR OFF

V: 0 = VERIFY OFF, 1 = VERIFY ON DESTINATION TRACK

X: STEPPING RATE LOW BIT

Y: STEPPING RATE HIGH BIT

	X	Y	RATE
WD1772 STEP RATES ARE:	0	0	2 MS
	0	1	3 MS
	1	0	6 MS
	1	1	12 MS

U: SET UP THE TRACK REGISTER TO TRACK FROM CURRENT HEADER (0 = NO, 1 = YES)

M: 0 = READ ONE SECTOR ONLY, 1 = READ MULTIPLE SECTORS

A: 0 = WRITE NORMAL DATA MARK, 1 = WRITE DELETED SECTOR MARK (NORMAL MEANS A VALID SECTOR, OTHERWISE IT'S ERASED)

E: 0 = NO HEAD SETTLING TIME, 1 = 30 MS DELAY TO SETTLE

© Copyright 1988 D. W. Martin All Rights Reserved.

P: 0 = ENABLE WRITE PRECOMPENSATION, 1 = DISABLE WRITE PRECOMPENSATION

I thru L are the WD177X controller interrupt condition.

I: DON'T CARE

J: DON'T CARE

K: INTERRUPT OCCURS WHEN INDEX HOLE IS ENCOUNTERED

L: IMMEDIATE INTERRUPT REGARDLESS OF CONDITION

When I thru L are all zero then the current command is terminated without an interrupt.

* * *

WD177X Status Byte

BIT	DESCRIPTION
0	BUSY FLAG 1 = COMMAND IS UNDER EXECUTION 0 = NO COMMAND IS EXECUTING
1	DATA REQUEST/INDEX 1 = DATA IS READY IN THE DATA REGISTER AT $6003. WHEN READING THIS MEANS THAT THE DATA REGISTER IS FULL, WHEN WRITING THIS MEANS THAT THE DATA REGISTER IS EMPTY. 0 = ALWAYS RESET TO ZERO WHEN UPDATED
2	LOST DATA/TRACK 00 TYPE 1 COMMANDS: INDICATES THAT THE HEAD IS ON TRACK 0, BIT 2 = 1 OTHER COMMANDS: INDICATES THAT THE DATA IN THE REGISTER AT $6003 WAS NOT READ OR WRITTEN IN TIME BY THE PROGRAM

© Copyright 1988 D. W. Martin All Rights Reserved.

3 CRC ERROR

CHECKSUM BYTES OF THE HEADER OR DATA BLOCK DECODED IN ERROR (BAD HEADER OR DATA)

4 RECORD NOT FOUND

SET TO ONE MEANS THAT THE DESIRED TRACK, SECTOR, OR SIDE WERE NOT FOUND

5 RECORD TYPE/SPIN-UP

TYPE 1 COMMANDS: INDICATED THAT THE MOTOR SPIN-UP SEQUENCE WAS COMPLETED

TYPE 2 AND 3 COMMANDS: THIS BIT INDICATES THE RECORD TYPE (0 = DATA MARK, 1 = DELETED DATA MARK)

6 WRITE PROTECT

WHEN SET TO ONE THIS MEANS THAT WRITE PROTECT IS ON (WRITE OPERATIONS ONLY) AND DISK YOU CANNOT WRITE TO THE CURRENT DISKETTE

7 MOTOR ON

STATUS OF THE DRIVE MOTOR (0 = MOTOR OFF, 1 = MOTOR ON)

* * *

Programming the WD177X

Type I Commands

RESTORE (SEEK TRACK 0)

Positions the read/write head to track zero.

SETUP: None.

RESULTS: Sets seek error status bit (if the V flag is set).

© Copyright 1988 D. W. Martin All Rights Reserved.

SEEK

Finds a specified track by stepping the head to the track being sought.

SETUP: Track register must contain the current track position of the R/W head and the data register contains the destination track position.

RESULT: Moves the read/write head in the appropriate direction until both registers are equal (desired track location reached) and an interrupt is generated.

STEP

Steps the read/write head in the same direction as the last step operation.

SETUP: None.

RESULT: If the U flag is on, the track register is updated. Upon completion of the command an interrupt is generated.

STEP-IN

Steps the read/write head in the direction towards track 79.

SETUP: None.

RESULT: If the U flag is on, the track register is incremented. Upon completion of the command an interrupt is generated.

STEP-OUT

Steps the read/write head in the direction towards track zero.

SETUP: None.

RESULT: If the U flag is on, the track register is decremented. Upon completion of the command an interrupt is generated.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Type II Commands

READ SECTOR

Reads a disk sector or sectors.

SETUP: Place the desired track and sector number in the track and sector registers.

RESULT: The disk drive attempts to read the selected sector or sectors from the disk. If the requested sector does not exist then the record not found status bit is set. Reading multiple sectors will be completed when the disk reaches the end of the current track or until the force interrupt command is loaded into the command register.

WRITE SECTOR

Write a disk sector or sectors.

SETUP: Place the desired track and sector number in the track and sector registers.

RESULT: The disk drive attempts to write the selected sector or sectors to the disk. If data is not ready or available then the load data status bit is set.

* * *

Type III Commands

READ ADDRESS

Reads the next id field that is encountered and transfers the data to the data register. The id field consists of: track number, side number, sector number, sector length, crc one and crc two. Sector sizes are coded as follows:

0 = 128 bytes
1 = 256 bytes
2 = 512 bytes
3 = 1024 bytes

SETUP: None.

RESULTS: At the end of the commands operation an interrupt is generated and the busy status bit is reset.

© Copyright 1988 D. W. Martin All Rights Reserved.

READ TRACK

Reads a track from the leading edge of the first index pulse and continues until the next index pulse. This command sends all gap, header and data bytes to the data register. The accumulation of bytes is synchronized to each address mark found.

According to the WD177X specifications this command can be useful for diagnostic purposes. Since, no CRC checking is done, gap information is included in the data, and the address mark detector is on while the command executes. The specs also mention that write splices or noise may cause the WD177X to look for an address mark.

NOTE: The ID address mark, ID field, ID CRC bytes, data address mark, data and data CRC bytes for each sector will be correct. Gap bytes may be read incorrectly during write splice time because of synchronization.

SETUP: Position the read/write head over the desired track (the one to read). Then issue the read track command.

RESULTS: An interrupt is generated upon command completion.

WRITE TRACK

Format diskettes on a per track basis.

SETUP: Position the read/write head over the desired track (the one to format). Then issue the write track command.

RESULTS: Writing starts at the leading edge of the first index pulse encountered and continues until the next index pulse. An interrupt is then generated. If data is not loaded into the data register in time the WD177X may set the lost data status bit and then generate an interrupt.

Sometimes the WD177X will use zeros as a substitute is data is not present in the data register. Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, and 1024 bytes.

© Copyright 1988 D. W. Martin All Rights Reserved.

The following chart lists special bytes used by the formatting process.

BYTE IN DATA REGISTER	MFM MEANING
00 to F4	Write 0 to F4 in MFM
F5	Write A1* in MFM and present CRC
F6	Write C2* in MFM
F7	Generate two CRC bytes
F8 to FB	Write F8 to FB in MFM
FC	Write FC in MFM
FD	Write FD in MFM
FE	Write FE in MFM
FF	Write FF in MFM

* * *

Type IV Commands

FORCE INTERRUPT

Used to stop the current command being executed by the WD177X (read and write commands) and it can be used to insure type I status in the status register.

SETUP: None.

RESULTS: Terminates the current command and resets the busy status bit.

* * *

WD177X Notes

The WD177X controller used in the 1581 is the WD1772 floppy disk controller chip. This chip must be accessed by machine language programs within the disk drive's memory.

If you are planning to program the WD1772 then I highly recommend the information that is available from Western Digital. They have published a small

© Copyright 1988 D. W. Martin All Rights Reserved.

booklet that gives details about the WD1772 covering hardware and programming.

See chapter 9 for information on how you can get the WD1772 booklet from Western Digital.

NOTE: Some early 1581's contained the WD1770 controller. If you have one of these drives it is recommended that you replace the WD1770 with a WD1772. Chapter 9 documents some known hardware bugs in the 1581 and Appendix A tells you about a program that will check for them.

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8000-$8001	CHECKSUM	CHECKSUM USED BY ROUTINES TO VERIFY THE INTEGRITY OF THE DOS ROMS
$8002-$8003		NOT USED BY THE DOS
$8004-	EXECMD	EXECUTE COMMAND STRING
$804C-	ENDCMD	END OF COMPUTER COMMAND GENERATE AN ERROR MESSAGE
$8050-	END0	END COMMAND BUT DON'T WRITE BAM
$805B-	END1	END COMMAND IGNORE ERROR
$8067-	END2	END COMMAND NO ERROR MESSAGE PREPARED
$8071-	CLRCMD	CLEAR COMMAND BUFFER
$8085-	SCAN	LOOK FOR ":" AND DRIVE NUMBER IN COMMAND STRING
$8099-	SCANCOLON	LOOK FOR COLON IN COMMAND STRING
$80A2-	TEST2FILES	TEST COMMAND WITH TWO FILENAMES FOR SYNTAX
$811C-	SCANCHARINA	SEARCH INPUT LINE FOR CHARACTER IN THE ACCUMULATOR
$8165-	LOOKCMD	SET ALL FLAGS AND LOOK AT COMMAND STRING TABLE
$81AF-	CLRCMDVAR	CLEAR AND SET BACK TABLE, POINTERS AND FLAGS
$81E5-	FLASHOFF	FLASH ERROR LED OFF
$81F1-	FLASHON	FLASH ERROR LED ON

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$81FD-	GETDRV	GET DRIVE NUMBER AND SET INTO FILE TABLE
$8224-	GETDRVFROMCMD	GET DRIVE NUMBER FROM COMMAND STRING
$8251-	INITWITHLEDON	INITIALIZE DRIVE AND SWITCH LED ON
$8270-	SETFILE	SET AND DETERMINE FILE TYPE
$8295-	CHKDRV	TEST VALID DRIVE NUMBER
$82A2-	INTDRVFNAME	INITIALIZE DRIVE GIVEN IN FILENAME
$82B9-	FINDFILE	LOOK FOR FILE ENTRY IN THE DIRECTORY
$8327-	SCANDIR	SEARCH DIRECTORY ENTRY
$8336-	SCANNEXT	SEARCH NEXT ENTRY
$83D7-	NEWSEARCH	REINITIALIZE FILE SEARCH FLAGS
$83E2-	QUITSEARCH	QUIT SEARCH FOR FILENAME
$83FA-	WCARD	1581 EXTENDED WILD CARD CHECK
$8424-	SETSEARCH	SET INDICATOR TO SEARCH IN DIRECTORY
$84AE-	INTDSK	INITIALIZE DISKETTE
$84EE-	WRTNAME	COPY FILENAME FROM INPUT BUFFER TO DIRECTORY BUFFER
$8508-	COPYDATA	COPY PART OF THE INPUT BUFFER AND THE CURRENT DATA BUFFER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8526-	FNAMELEN	SEARCH LENGTH OF FILENAME IN INPUT BUFFER (STARTING POSITION IN .X)
$854D-	READFROMDIR	READ FILE FROM DIRECTORY
$855D-	DIROUTPUT	ESTABLISH DIRECTORY FOR OUTPUT TO BUFFER
$861C-	DELBUF	DELETE BUFFER FOR DATA NAME WITH EMPTY CHARACTER
$868B-	BLKFRE	SETUP CLOSING LINE OF DIRECTORY WITH "BLOCKS FREE." MESSAGE
$867C-$8687	BLKFREMSG	"BLOCKS FREE." MESSAGE
$8688-	SCRATCH	SCRATCH COMMAND - "S"
$86DB-	FRESIDE	FREE SIDE SECTOR BLOCKS IN A REL FILE
$8713-	FREEUP	PURSUE SECTORS ON HAND AND FREE UP IN BAM
$8732-	FREEUP1	FREE SECTOR IN BAM AND CONTINUE
$873B-	DELFILE	FILE ENTRY IN FILE TYPE OF DIRECTORY MARKED AS SCRATCHED
$8746-	NEW	ROUTINE FOR 1581 "NEW" COMMAND (FORMAT DISK) - "N"
$876E-	COPY	COPY COMMAND FOR COPYING FILES - "C"
$879B-	COPYFILE	COPY FILES
$87F4-	COPYSING	COPY SINGLE FILES
$8800-	COPYMULT	COPY MULTIPLE FILES

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8841-	OPENREAD	OPEN CHANNEL TO READ FILE
$8876-	GETBYTE	READ A BYTE FROM A FILE
$8895-	COPYREL	COPY A RELATIVE FILE
$88C5-	RENAME	RENAME A FILE COMMAND- "R"
$8903-	CHKEXIST	SEE IF FILE ENTRY EXISTS
$891E-	CMPNAMES	COMPARE WITH TWO FILENAMES
$892F-	MEMCMD	MEMORY COMMAND ROUTINES
$8954-	MEMREAD	MEMORY READ COMMAND GET A BYTE FROM DRIVE MEMORY - "M-R"
$8983-	MEMWRT	MEMORY WRITE COMMAND WRITE A BYTE INTO DRIVE MEMORY - "M-W"
$898F-	USER	USER COMMANDS TO START PROGRAMS IN DOS BUFFER AT $0500
$8996-	BURSTCMD	ROUTINE FOR BURST USER COMMANDS ("U0")
$89CC-	EXECUSER	EXECUTE A USER COMMAND
$89E4-	DIRECT	"#" COMMAND - OPEN A DIRECT CHANNEL
$8A5D-	BLKCMDS	BLOCK COMMANDS
$8A9F-	BLKPARAM	GET AND SET BLOCK COMMAND PARAMETERS
$8AAC-	TESTPARAM	TEST BLOCK COMMAND PARAMETERS
$8AD0-	ASCII2BIN	CONVERT/SET BLOCK COMMAND PARAMETERS FROM ASCII TO BINARY

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8B20-	BINVAL	BINARY VALUE TABLE $01, $0A, $64
$8B23-	BLKFRE	BLOCK FREE COMMAND- FREE A BLOCK IN THE BAM - "B-F"
$8B2F-	BLKALLOC	BLOCK ALLOCATE COMMAND - MARK A BLOCK IN THE BAM AS USED - "B-A"
$8B65-	TESTBR	TEST BLOCK READ ("B-R") PARAMETERS AND READ SECTOR INTO BUFFER
$8B6B-	GETBUFBYTE	GET BYTE FROM BUFFER
$8B71-	READSECINTPTR	READ SECTOR FROM DISKETTE TO BUFFER AND INITIALIZE POINTER
$8B85-	BLKREAD	READ SECTOR FROM DISKETTE FOR "B-R" COMMAND
$8B8E-	BREXTEND	BLOCK SHIFT READ COMMAND - EXTENDED TRACK READER
$8B9A-	USER1CMD	ROUTINE FOR "U1" COMMAND READ SECTOR FROM DISKETTE
$8BAE-	BLKWRT	ROUTINE FOR BLOCK WRITE COMMAND "B-W"
$8BD1-	BWEXTEND	BLOCK SHIFT WRITE COMMAND - EXTENDED TRACK WRITER
$8BD7-	USER2CMD	ROUTINE FOR "U2" COMMAND WRITE SECTOR TO DISKETTE
$8BE3-	BLKEXEC	ROUTINE FOR BLOCK EXECUTE COMMAND READ SECTOR AND EXECUTE CODE IN DOS BUFFER "B-E"

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8BFA-	BLKPTR	ROUTINE FOR BLOCK POINTER COMMAND SET BUFFER POINTER ("B-P")
$8C0F-	ALLOCOPEN	ALLOCATE BUFFER AND OPEN CHANNEL
$8C2F-	CHKPARAM	TEST PARAMETERS FOR VALID SECTOR ASSIGNMENT
$8C44-	SKIPILLEGAL	SAME AS ABOVE, BUT DOES NOT FLAG ILLEGAL TRACKS AND/OR SECTORS
$8C5C-	ALLOCBUF	ALLOCATE RAM BUFFER- .A CONTAINS THE BUFFER NUMBER (1 = BUFFER 0, 2 = BUFFER 2, ETC.)
$8C61-$8C6A	BLKCMDTAB	BLOCK COMMAND TABLE "AFRWEPRW?*"
$8C6B-$8C7E	BLKADDR	ADDRESSES OF 10 BLOCK COMMANDS IN LOW, HIGH BYTE FORMAT $8B2F - "B-A" $8B23 - "B-F" $8B85 - "B-R" $8BAE - "B-W" $8BE3 - "B-E" $8BFA - "B-P" $8B8E - "B-R" $8BD1 - "B-W" $8C7F - "B-?" $8C84 - "B-*"
$8C7F-	AUTHOR	BLOCK-? COMMAND AUTHOR/DESIGNER MESSAGE IN ERROR CHANNEL - "B-?"
$8C84-	DEDICATE	BLOCK-* COMMAND DEDICATION MESSAGE IN ERROR CHANNEL - "B-*"
$8C89-	GETREC	GET RECORD FROM RELATIVE FILE

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8CC1-	NUMBYTES	COMPUTE NUMBER OF BYTES UP TO RECORD
$8D06-	DIV254	DIVISION OF MATH REGISTER BY 254 (SECTOR LENGTH)
$8D09-	DIV120	DIVISION OF MATH REGISTER BY 120 (RECORD ENTRIES IN SIDE-SECTOR)
$8D38-	CLRMATH1	CLEAR MATH REGISTER 1
$8D41-	MULTTIMES4	MULTIPLY MATH REGISTER 2 FOUR TIMES
$8D44-	DOUBLEREG2	DOUBLE MATH REGISTER 2
$8D4C-	ADD1TO2	ADD MATH REGISTER 2 TO MATH REGISTER 1
$8D59-	INTBUFCHAN	INITIALIZE BUFFER CHANNEL TABLE
$8D68-	TESTCHANNUM	TEST CHANNEL NUMBER IN BUFFER CHANNEL TABLE
$8D7D-	MAMBUF	MANAGE AND ASSIGN BUFFERS
$8E3C-	FINDFREBUF	LOOK FOR A FREE BUFFER
$8E4D-	SETBUFSTATUS	TOGGLE BUFFER FROM ACTIVE TO PASSIVE AND BACK AGAIN
$8E5C-	WRTINTERNAL	WRITE BYTES OVER INTERNAL CHANNEL INTO BUFFER
$8E78-	WRT2FILE	WRITE BYTE INTO FILE
$8EB1-	WRT2BUF	WRITE BYTE IN CURRENT BUFFER
$8EC5-	INITIALIZE	INITIALIZE COMMAND- "I0"

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$8FD6-	READ2BUF	READ SECTOR FROM DISKETTE TO BUFFER
$8FEA-	READNEXT	READ IN GIVEN SECTOR AND SECTOR AFTER THAT
$8FFE-	READSEC	READ SECTOR FROM DISK
$9002-	WRTSEC	WRITE SECTOR TO DISK
$9027-	OPENREAD	OPEN CHANNEL FOR READING
$9042-	SCANANDOPEN	SEARCH FOR AND OPEN CHANNEL
$905F-	GETFILETYPE	GET CURRENT FILE TYPE
$9069-	GETCHANANDBUFF	GET CHANNEL AND MATCHING BUFFER NUMBER
$9071-	GETFROMBUF	GET BYTE FROM CURRENT BUFFER
$909B-	GETFROMFILE	GET BYTE FROM FILE
$9112-	WRT2FILE	WRITE BYTE IN FILE
$9138-	NEXTCHAR	SET CURRENT BUFFER POINTER TO NEXT CHARACTER
$9145-	SWITCHAUTO	SWITCH FOR AUTOLOADER BOOT ON INITIALIZE OR BURST INQUIRY/QUERY
$9157-	SCANWRTCHAN	LOOK FOR WRITE CHANNEL AND BUFFER
$915A-	SCANREADCHAN	LOOK FOR READ CHANNEL AND BUFFER
$919E-	FRECHAN	FREE UP CHANNEL
$91CE-	FREBUFCHAN	FREE UP BUFFER AND CORRESPONDING CHANNEL
$9204-	SCAN4BUF	LOOK FOR BUFFER
$9228-	SCAN4FREBUF	LOOK FOR FREE BUFFER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$923E-	FREINACT	FREE UP ALL INACTIVE BUFFERS
$9252-	FREEINDEX	FREE UP BUFFER INDEX
$9262-	CLOSECHAN	CLOSE CHANNELS 0 TO 14
$926E-	FREEALLCHAN	FREE UP ALL CHANNELS ON CURRENT DRIVE
$9291-	GETBUFF	GET A BUFFER
$92DB-	LAYOUTFREE	SEEK AND LAYOUT A FREE CHANNEL
$92F4-	GETFROMCHAN	GET A BYTE FROM A CHANNEL
$9303-	READFROMFILE	READ BYTE FROM A FILE
$933A-	READFROMREL	GET BYTE FROM A RELATIVE FILE
$9348-	GETNEXT	GET NEXT BYTE FROM FILE
$934A-	GETCURREN	GET CURRENT BYTE FROM FILE
$9370-	READERRCHAN	READ ERROR CHANNEL
$9396-	ERRPTR	SET POINTER FOR ERROR MESSAGE POINTER
$939F-	INITERRCHAN	INITIALIZE ERROR MESSAGE CHANNEL
$93AA-	READNXTSEC	READ NEXT SECTOR OF A FILE
$93B0-	JOBREAD	TAKE JOB CODE FOR READ SECTOR ($80)
$93C1-	JOBWRT	TAKE JOB CODE FOR WRITE SECTOR ($90)
$93CF-	OPENSEQREAD	OPEN SEQUENTIAL FILE FOR READING
$93E0-	OPENSEQWRITE	OPEN FILE FOR WRITING

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$93E7-	WRTNEXTDIR	WRITE NEXT DIRECTORY SECTOR
$9422-	SETBUFPTR1	SET BUFFER POINTER TO GIVEN POSITION
$9434-	CLOSEINTERNAL	CLOSE INTERNAL CHANNELS
$9442-	CURRENTBUF	DETERMINE CURRENT BUFFER POINTER
$9445-	SETBUFPTR2	SET BUFFER POINTER (BUFFER NUMBER IN .A)
$9450-	GETBUFBYTE	READ ANY BYTE FROM BUFFER (.A MUST CONTAIN POSITION OF THE CHARACTER)
$9460-	CHKTRKSEC	TEST FOR VALID TRACK AND SECTOR NUMBERS THEN SET JOB CODE
$94A8-	GETTRKSEC	GET TRACK AND SECTOR OF CURRENT JOB FROM JOB MEMORY
$94B5-	RANGECHK	CHECK CURRENT TRACK AND SECTOR FOR ALLOWABLE RANGE
$94CB-	FALSEFORMAT	DISPLAY ERROR MESSAGE FOR FALSE FORMAT
$94D3-	SENDJOBCURBUF	SEND JOB FOR CURRENT BUFFER TO JOB LOOP
$94DE-	READNWAIT	SEND JOB CODE FOR READ TO JOB LOOP AND WAIT UNTIL EXECUTION
$94E2-	WRTNWAIT	SAME AS ABOVE EXCEPT WRITE
$94E4-	EXECJOB	EXECUTE JOB FOR CURRENT DRIVE (JOB CODE IN .A)
$94E6-	EXECJOB2	EXECUTE JOB CODE (JOB CODE IN .A, BUFFER NUMBER IN .X)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$94E8-	EXECJOB3	EXECUTE JOB
$94ED-	JOBDONE	WAIT UNTIL JOB IS EXECUTED AND AN ERROR MESSAGE IS PREPARED
$94F8-	SUPERVISE	SUPERVISE THE CURRENT JOB RUN
$951A-	NXTRKONERR	SET HEAD TO NEXT TRACK AFTER A READ ERROR-SEARCH SOME MORE
$9564-	WAITTILDONE	JOBCODE EXECUTES UNTIL SUCCESSFUL OR UNTIL COUNTER IN $30 = 0
$9585-	SENDCURRENT	SEND CURRENT TRACK AND SECTOR NUMBERS TO JOB LOOP
$9588-	SENDTRKSEC	SEND TRACK AND SECTOR NUMBERS TO JOB LOOP (BUFFER IN .A)
$959D-	DIRECTCALL	DIRECT 1581 CONTROLLER CALL
$95AB-	CLOSEFILE	CLOSE A FILE ENTRY IN THE DIRECTORY
$9678-	OPENCMD	TAKE ON OPEN COMMAND WITH A SECONDARY ADDRESS 0 TO 14
$97A2-	SAVEREPLACE	OVER WRITE CORRESPONDING FILE ENTRY
$984D-	OPENREADFILE	OPEN A FILE FOR READING
$9890-	OPENWRTFILE	OPEN A FILE FOR WRITING
$98AB-	SETCMD	SETUP FILE TYPE AND FILE OPERATION AS COMMAND STRING
$98CC-	APPEND2FILE	PREPARE FILE FOR APPEND

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$98F7-	XMITDIR	TRANSMIT DIRECTORY TO COMPUTER
$995C-	CLOSEAFILE	CLOSE A FILE
$9986-	CLOSEALL	CLOSE ALL FILES
$999F-	CLOSE2ND	FILES DECLARED THROUGH SECONDARY ADDRESS CLOSED
$9A2A-	WRTLASTSEC	WRITE THE LAST SECTOR OF A FILE TO DISKETTE
$9A72-	CLOSEWRT	CLOSE DIRECTORY ENTRY AFTER WRITE OPERATION
$9B0D-	OPENREADCHAN	OPEN CHANNEL TO READ FILE
$9B9B-	INITOPENPTR	INITIALIZE CHANNEL OPEN POINTER
$9BC3-	OPENWRTCHAN	OPEN CHANNEL TO WRITE TO A FILE
$9C82-	SETRELSS	SETUP A REL FILE SIDE SECTOR
$9CCA-	WRT2SS	WRITE A BYTE TO CURRENT SIDE SECTOR
$9CD3-	CHANINFT	CHANNEL NUMBER IN FILE TYPE FLAG SET (CARRY = 1) OR CLEARED (CARRY = 0)
$9CD5-	CHFT1	VALUE COMBINED IN FILE TYPE (BIT = 1 IS SET)
$9CDB-	CHFT2	REMOVE VALUE FROM THE FILE TYPE FLAG (BIT = 1 IS TAKEN OUT/NOT SET)
$9CE4-	CHFT3	CHECK FOR SET FILE TYPE FLAG (THE FLAG VALUE IS IN .A)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$9CE9-	CHFT4	CHECK TO SEE IF JOB CODE IS SETUP FOR WRITING
$9CF5-	TESTFPTR	TEST FILE POINTER
$9D2E-	BUF2DSK	WRITE BUFFER TO DISK
$9D3A-	SETCHAIN	SET CHAINED BYTES WHICH POINT TO THE NEXT SECTOR
$9D49-	GETCHAIN	GET LINKED BYTES WHICH POINT TO THE NEXT SECTOR
$9D56-	SETENDLNK	SET SECTOR LINK BYTES AS LAST SECTOR IN CHAIN OF LINKED BYTES AND/OR SECTORS
$9D69-	BUFPTR0	SET CURRENT BUFFER POINTER TO ZERO
$9D79-	GETCURTRKSEC	GET CURRENT TRACK AND SECTOR OF CURRENT JOB/GET CHAN OF CURRENT SECONDARY ADDRESS
$9D7C-	GETCUR2	GET TRACK AND SECTOR OF CURRENT JOB/DETERMINE BUFFER
$9D8E-	SENDJOB	GIVE JOB CODES TO JOB LOOP
$9DCE-	NXTLINK	NEXT SECTORS PARAMETERS SET BY LINKED BYTES THAT ARE ON HAND
$9DDE-	BUFCOPY	COPY FILE FROM ONE BUFFER TO ANOTHER BUFFER. THE .A CONTAINS THE NUMBER OF BYTES TO TRANSFER, .YR IS THE SOURCE BUFFER NUMBER, AND .XR IS THE DESTINATION BUFFER NUMBER.

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$9DFA-	CLRBUFF	CLEAR THE BUFFER NUMBER IN .A WITH ZEROS
$9E0B-	SETSSNUM	GET THE NUMBER OF THE CURRENT SIDE SECTOR
$9E15-	SETBUFFPTR	SET THE BUFFER POINTERS $64/$65 TO ANY POSITION IN THE BUFFER
$9E23-	SETBUFFPTR	SET THE BUFFER POINTER
$9E32-	READSIDE	READ A SIDE SECTOR INTO A BUFFER AND SETUP POINTERS
$9E56-	READSEC	READ A SECTOR- THE BUFFER POINTER OF THE CURRENT BUFFER MUST USE THE TRACK AND SECTOR PARAMETERS OF THE LINK BYTES
$9E75-	SETSSPTR	SET THE SIDE SECTOR POINTER
$9E7D-	CALCNUMSS	CALCULATE THE NUMBER OF SIDE SECTORS IN A RELATIVE FILE
$9EE4-	SSSTATUS	TEST STATUS OF A SIDE SECTOR
$9F11-	CURBUF	DETERMINE NUMBER OF THE CURRENT BUFFER
$9F1C-	CURBUFST	GET CURRENT BUFFER STATUS
$9F33-	BUFFREORNOT	TEST WHETHER BUFFER IS FREE
$9F3E-	TWOBUFF	ACTIVATE BUFFERS FOR TWO BUFFER OPERATIONS
$9F4C-	WRTREC	WRITE A RECORD FOR A RELATIVE FILE
$9FB6-	PTR2LAST	SET POINTER TO LAST CHARACTER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$9FBF-	PREPRECSEC	PREPARE SECTOR OF THE RECORD
$9FFC-	WRTRECCHAR	WRITE A CHARACTER OF THE RECORD INTO THE BUFFER
$A033-	WRTREC2DATABUF	WRITE RECORD TO THE DATA BUFFER
$A07B-	FILREC	FILL THE REST OF A RECORD WITH EMPTY BYTES ($00/0)
$A08D-	DATAALT	FLAG FOR BUFFER DATA ALTERED IS SET
$A09C-	DATANOTALT	FLAG FOR BUFFER DATA ALTERED IS CLEARED
$A0A6-	GETFROMREC	GET A BYTE FROM A RECORD
$A0E1-	OUTPUTAREC	GET A RECORD AND OUTPUT IT
$A0EC-	RECNOTHERE	RECORD NOT PRESENT ERROR
$A0FD-	LSTRECCHAR	SET POINTER TO LAST CHARACTER OF THE RECORD
$A143-	FINDENDREC	SEARCH FOR THE END OF A RECORD
$A15C-	FINDENDREL	SEARCH FOR THE END OF A RELATIVE FILE
$A1A1-	RECORD	RECORD COMMAND ROUTINE ("P")
$A20D-	READREC	READ A RECORD INTO THE BUFFER
$A235-	READRECSEC	READ THE RECORD SECTOR CONTAINED IN THE BUFFER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$A273-	CHKSEC	CHECK TO SEE IF SECTOR IS ALREADY IN THE BUFFER
$A298-	ADDREC	ENTER A NEW RECORD IN THE SECTOR
$A2BC-	CALCRECPOS	CALCULATE POSITION OF A NEW RECORD IN THE SECTOR
$A2D6-	INSERTNEWREC	INSERT A NEW RECORD IN A RELATIVE FILE
$A459-	NEWSS	PREPARE A NEW SIDE SECTOR
$A547-	SUPERSS	ROUTINES FOR HANDLING SUPER SIDE SECTORS
$A602-$A730	ERRMSGS	ERROR MESSAGES STORED AS ASCII TEXT

	ERROR NUMBER		TEXT
$A602-	E00	00	"OK"
$A606-	E02	02	"PARTITION SELECTED"
$A61A-	E20ETC	20-24,27	"READ ERROR"
$A625-	E52	52	"FILE TOO LARGE"
$A631-	E50	50	"RECORD NOT PRESENT"
$A63C-	E51	51	"OVERFLOW IN RECORD"
$A649-	E25	25	"WRITE ERROR"
$A64D-	E26	26	"WRITE PROTECT ON"
$A65A-	E29	29	"DISK ID MISMATCH"

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL		DESCRIPTION
	ERROR NUMBER		TEXT
$A660-	E30ETC	30-34	" S Y N T A X ERROR"
$A66C-	E60	60	"WRITE FILE OPEN"
$A670-	E63	63	"FILE EXISTS"
$A679-	E64	64	"FILE TYPE MISMATCH"
$A681-	E65	65	"NO BLOCK"
$A68A-	E66ETC	66-67	" I L L E G A L TRACK OR SECTOR"
$A6A3-	E61	61	"FILE NOT OPEN"
$A6A7-	E39/62	39,62	"FILE NOT FOUND"
$A6AC-	E01	01	"FILES SCRATCHED"
$A6B9-	E70	70	"NO CHANNEL"
$A6C4-	E71	71	"DIRECTORY ERROR"
$A6C9-	E72	72	"DISK FULL"
$A6D0-	E73	73	"COPYRIGHT CBM DOS V10 1581"
$A6EB-	E74	74	"DRIVE NOT READY"
$A6F8-	E75	75	" F O R M A T ERROR"
$A705-	E76	76	"CONTROLLER ERROR"

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
	ERROR NUMBER	TEXT
$A716-	E77	77 "SELECTED PARTITION ILLEGAL"
$A731-	E79	79 "SOFTWARE BY D A V I D SIRACUSA. HARDWARE BY G R E G BERLIN"
$A75F-	E7A	7A "DEDICATED TO MY WIFE LISA"

$A779-$A7AD ERRTOK ERROR MESSAGE TOKENS

	TOKEN NUMBER	TEXT
$A779-	T09	09 "ERROR"
$A77F-	T0A	0A "WRITE"
$A785-	T03	03 "FILE"
$A78A-	T04	04 "OPEN"
$A78F-	T05	05 "MISMATCH"
$A798-	T06	06 "NOT"
$A79C-	T07	07 "FOUND"
$A7A2-	T08	08 "DISK"
$A7A7-	T0B	0B "RECORD"

$A7AE-	DOERR	ERROR MESSAGE OUTPUT ROUTINE .A MUST CONTAIN THE ERROR NUMBER, .XR THE BUFFER NUMBER
$A7F1-	PREPMSG	PREPARE THE ERROR MESSAGE
$A7F4-	ACTMSG	ACTIVATE ERROR MESSAGE
$A83E-	BIN2BCD	CONVERT A BINARY NUMBER TO A BCD NUMBER
$A850-	BCD2ASCII	CONVERT A BCD NUMBER INTO TWO ASCII CHARACTERS

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$A862-	OKMSG	PREPARE "00, OK" ERROR MESSAGE
$A867-	ERRTRKSEC	OUTPUT ERROR MESSAGE WITH TRACK AND SECTOR NUMBERS = 0
$A86D-	ERRBUF	PRODUCE ERROR MESSAGE IN BUFFER (BUFFER NUMBER IN .A)
$A8AD-	WRTMSG2BUF	WRITE ERROR MESSAGE IN TEXT FORM TO ERROR BUFFER
$A8F8-	WRTASCIIMSG	WRITE ASCII CHARACTERS INTO A BUFFER. NON-ASCII CHARACTERS INTERPETED AS ERROR NUMBERS
$A90E-	ERRMSGFROMROM	GET A CHARACTER OF ERROR TEXT FROM THE ERROR MESSAGE TEXT TABLE IN ROM
$A91C-	GETBYTETAB	GET THE CURRENT BYTE FROM THE TABLE
$A926-$A937	AUTOFILENAME	AUTO EXECUTE FILE NAME. FIRST CHARACTER IS THE "&" COMMAND "&COPYRIGHT CBM 86"
$A938-	AUTOLOADER	CBM AUTO LOADER ROUTINE
$A94C-	DISABLEAUTO	RETURN FROM THE AUTO LOADER WITH THE AUTO LOADER DISABLED
$A956-	UTILITYCMD	UTILITY LOADER COMMAND "&" FIND AND GET UTILITIY LOADER PROGRAM BLOCK

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$A9F5-	GETUTL	READ IN A BYTE FROM UTILITY LOADER PROGRAM BLOCK
$AA07-	UTLCHKSUM	IMPLEMENT A CHECKSUM FOR UTILITY LOADER PROGRAM BLOCK
$AA0F-	SETERRVEC	SET ERROR VECTORS TO ROUTINE AT $A94C
$AA27-	INTERLEAVE	"U0>SX" COMMAND SET SECTOR FORMAT FOR CBM DISKETTES (INTERLEAVE)
$AA2D-	READATTEMPT	"U0>RX" COMMAND SET NUMBER OF READ ATTEMPTS
$AA33-	SIEEETIMING	"U0>IX" COMMAND FUNCTION SET SIEEE TIMING
$AA39-	TESTROM	"U0>T" COMMAND TEST ROM CHECKSUM AKA ROM SIGNATURE ANALYSIS
$AA3C-	BURSTUTL	DECODE AND EXECUTE DRIVE STATUS AND CONTROL FUNCTIONS
$AA65-	SETDEVICE	SET DEVICE NUMBER VIA "U0>"+CHR$(X)
$AA83-	SYNTAXERR	SYNTAX ERROR MESSAGE
$AA88-	BUSMODE	"U0>BX" COMMAND SET SERIAL BUS MODE (SLOW OR FAST)
$AA9A-	VERIFYMODE	"U0>VX" COMMAND SETS VERIFY MODE SELECTION (ON OR OFF)
$AAA8-	BURSTMEMCMDS	BURST MEMORY READ AND WRITE COMMANDS
$AAC3-	BURSTMEMREAD	BURST MEMORY-READ COMMAND

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$AAD7-	BURSTMEMWRT	BURST MEMORY-WRITE COMMAND
$AB09-	SETVERIFY	CHECK VERIFY MODE SELECT VALUE EITHER ZERO OR ONE
$AB1D-	SIGNATURERTN	ROM SIGNATURE ANALYSIS ROUTINE TEST ROM VIA CHECKSUM
$ABCF-	ATN	ROUTINE FOR CONTROLLING THE SERIAL BUS (SERIAL BUS ATN SERVER)
$ACBB-	BUS2INPUT	SWITCH 1581 BUS TO INPUT
$ACD4-	BUS2OUTPUT	SWITCH 1581 BUS TO OUTPUT
$ACE8-	DATALOW	DATA LINE SET LOW
$ACF1-	DATAHI	DATA LINE SET HIGH
$ACFA-	CLOCKHI	CLOCK LINE SET HIGH
$AD03-	CLOCKLOW	CLOCK LINE SET LOW
$AD0C-	SERVAL	VALUES READ FROM SERIAL BUS
$AD2F-	DELAY1	CYCLE DELAY
$AD34-	DELAY2	CYCLE DELAY
$AD3C-	UICMD	"UI" COMMAND BUS MODE COMMAND 1541/1540 SPEED
$AD5C-	TALK	SERIAL BUS TALK ROUTINE
$AEB8-	LISTEN	SERIAL BUS LISTEN ROUTINE
$AED9-	RESETBUS	RESET BUS CONTROL REGISTER AND WAIT FOR NEXT COMMAND

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$AEEA-	SETSERIAL	SPINP/SPOUT SETUP FAST SERIAL DIRECTION AS INPUT OR OUTPUT (CARRY SET = SPOUT, CARRY CLEAR = SPINP)
$AEF2-	RAMORROMERR	RAM OR ROM ERROR (TEST/CHECKSUM)
$AF24-	TESTRAM&ROM	TEST THE 1581'S RAM AND ROM
$AFCA-	INITZPG	INITIALIZE ZERO PAGE
$AFDE-	PUPMSG	GENERATE DOS POWER UP MESSAGE
$B0B3-	INITLAYOUT	INITIALIZE DISK LAYOUT VARIABLES (MAX TRACK, DIR TRACK, ETC.)
$B0CF-	FORMATNORM	SET UP A "NORMAL" DOS FORMAT FOR BURST FORMAT COMMAND
$B0F0-	IDLE	MAIN IDLE LOOP
$B17C-	LOADDIR	LOAD DIRECTORY "$"
$B201-	DIREND	DIRECTORY OUTPUT ENDED
$B237-	COPYDIR	COPY DIRECTORY ENTRY INTO CURRENT BUFFER
$B245-	GETDIRBYTE	GET A BYTE FROM THE DIRECTORY
$B262-	VALIDATE	VALIDATE COMMAND ROUTINE
$B286-	REPAIRBAM	ALL BLOCKS OF A FILE PUT INTO BAM ALLOCATES BLOCKS ACCORDING TO FILE LINK BYTES
$B2C7-	TSTBLOCKS	ALL BLOCKS FOLLOWING A FILE ARE TESTED FOR VALIDITY

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$B2EF-	TSTILLEGAL	CHECK FOR ILLEGAL SYSTEM TRACK OR SECTORS
$B348-	NEW/FORMAT	FORMAT COMMAND (NEW)
$B390-	NEWBAM	CREATE A NEW BAM (ALL SECTORS FREE)
$B430-	CLRBAMBUF	CLEAR BAM BUFFERS
$B44A-	NEWBAM2	PRODUCE A NEW 1581 BAM FOR VALIDATE COMMAND
$B546-	FRESEC	SECTOR RELEASED AND MARKED AS FREE
$B572-	SECUSED	MARK A SECTOR IN THE BAM AS USED. IF NONE ARE FREE THEN A "DISK FULL" ERROR IS PRODUCED.
$B5B4-	DRVNOTREADY	PRODUCES "DRIVE NOT READY" ERROR
$B5D8-	BAMPTR	BAM BUFFER POINTER SET TO BIT FOR CURRENT SECTOR AND BIT RETRIEVED
$B5EA-	ISOLATEMASKS	MASKS TO ISOLATE BAM BITS
$B668-	SCANFREBLK	LOOK FOR NEXT FREE BLOCK IN THE BAM
$B6BF-	NXTFREONTRK	LOOK FOR NEXT FREE SECTOR ON THIS TRACK
$B6ED-	NXTOPTSEC	LAYOUT NEXT OPTIMUM SECTOR
$B75E-	NUMFREALL	CHECK NUMBER OF FREE BLOCKS IN BAM FOR EVERY TRACK
$B781-	PARTITION	COMMAND TO CREATE OR SWITCH PARTITIONS- "/"

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$B7F7-	SETSUBDIR	MOVE THROUGH SUB-DIRECTORIES TO ROOT DIRECTORY
$B888-	ILLEGALPART	AN ILLEGAL PARTITION WAS SELECTED
$B8D5-	FASTLOAD	FASTLOAD FILE OVER 1581 BUS (PRG, SEQ, USR)
$B95F-	XFERFAST	FAST SECTOR TRANSFER
$B990-	XFERLAST	FAST LOAD LAST FILE SECTOR
$B9D3-	SHOWERR	DISPLAY ERROR MESSAGES
$B9DF-	LOADERR	LOAD ERROR
$BA06-	FNAMESETUP	SHIFT FILE NAME TO BEGINNING OF INPUT BUFFER
$BA40-	SENDFAST	BYTE SENT OVER 1581 BUS FOR FAST LOAD
$BA64-	SETUPCRASH	THIS ROUTINE SETS THE RAM ERROR VECTORS TO POINT TO ROM. THE VECTOR AT $01BA IS SET TO $DFDF WHICH WILL CRASH THE DOS.
$BA7C-	SAVEVEC	STORE ERROR VECTORS IN SAVE VECTOR AREA
$BA95-	RESTOREVEC	RETRIEVE ERROR VECTORS FROM SAVE VECTOR AREA
$BAB3-	BURSTREAD	BURST READ TRACK AND SECTOR HANDLER
$BAD6-	BURSTREAD2	BURST READ NUMBER OF SECTORS HANDLING ROUTINE
$BAF7-	IDMISMATCH	DISK ID MISMATCH ERROR
$BAF9-	DRVNOTREADY	DRIVE NOT READY ERROR

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$BAFC-	COMMSTERROUT	COMBINE COMMAND STATUS FLAG AND OUTPUT WITH ERROR
$BB02-	EVENTERR	EVENTUAL ERROR OUTPUT (OTHERWISE RETURN)
$BB0A-	DOERRINXR	OUTPUT ERROR MESSAGE NUMBER (NUMBER IN .XR)
$BB11-	BURSTREADCMD	BURST READ COMMAND
$BC01-	BURSTWRTCMD	BURST WRITE COMMAND
$BCB2-	INQUIREDISK	BURST COMMAND INQUIRE DISK
$BD06-	MOREVALUES	VALUES $00, $10, $0A, $05
$BD0A-	DRVNOTREADY2	DRIVE NOT READY ERROR
$BD12-	BURSTFORMAT	BURST FORMAT COMMAND
$BD4A-$BD5D	FORMATSTRING	"N0:COPYRIGHT CBM,86" IN ASCII (USED TO DO 1581 DEFAULT BURST FORMAT)
$BD5E-	FORMATSTANDARD	FORMAT USING STANDARD DOS FORMAT VIA BURST FORMAT COMMAND
$BD7C-	CUSTOMFORMAT	CUSTOM FORMAT VIA BURST FORMAT COMMAND/SETUP FORMAT VARIABLES
$BDF8-	MOREVALUES2	VALUES $0E, $16, $26, $44
$BDFC-	SYNTAXERR	SYNTAX ERROR
$BE06-	QUERYDISK	BURST QUERY DISK FORMAT COMMAND.
$BE79-	SENDQUERY	SEND OUT THE RESULTS OF THE QUERY DISK FORMAT

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$BEBB-	INQUIRESTATUS	BURST INQUIRE STATUS COMMAND
$BEF1-	SETSTATUS	SET COMMAND STATUS BYTE
$BEF8-	SYNTAXERR	SYNTAX ERROR
$BF02-	DUMPCACHE	BURST COMMAND DUMP TRACK CACHE BUFFER
$BF66-	PREPERROUT	PREPARE ERROR BYTE OUTPUT
$BF7F-	PREPDATA	VALUES $00, $10, $20, $30
$BF86-	SENDBYTEFAST	SEND BYTE OVER SERIAL BUS USING BURST PROTOCOL
$BFE3-	DUMPTRK	DUMP A TRACK FROM CACHE BUFFER TO DISK
$C097-	SMTOGRT	DETERMINE SMALLEST AND GREATEST SECTOR NUMBERS
$C0BE-	JMPCTRLER	JUMP TO THE DISK CONTROLLER ROUTINE
$C163-$C183	CTRLBYTES	DRIVE CONTROLLER BYTES/CODES CONTAINING 8 FLAG BITS FOR EACH OF THE 33 AVAILABLE JOBS

BIT	TEXT
7	TRANSLATE TRACK AND SECTOR
6	CACHE OUT
5	START MOTOR
4	WAIT UNTIL MOTOR IS UP TO SPEED
3	SEEK TRACK
2	CHECK CACHE DIRTY
1	STEP REQUIRED
0	SIDE SELECT

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$C184-$C1A4	CTRLBYTES2	DRIVE CONTROLLER BYTES/CODES CONTAINING 3 MORE FLAG BITS FOR EACH OF THE 33 AVAILABLE JOBS

BIT	TEXT
7	BY-PASS TRACK AND SECTOR TRANSLATION
6	LOGICAL/PHYSICAL FLAG
5	READ/WRITE OPERATOR

$C1A5-$C1E7 JOBCMDS — JOB QUEUE COMMAND VECTORS

	NAME	COMMAND NUMBER	
$C1A5-	READ_DV	$80	$C900
$C1A7-	RESET_DV	$82	$C2E7
$C1A9-	MOTON_DV	$84	$C390
$C1AB-	MOTOFF_DV	$86	$C393
$C1AD-	MOTONI_DV	$88	$C396
$C1AF-	MOTOFFI_DV	$8A	$C3A9
$C1B1-	SEEK_DV	$8C	$C3AF
$C1B3-	FORMAT_DV	$8E	$C3BB
$C1B5-	WRSTD_DV	$90	$C900
$C1B7-	DISKIN_DV	$92	$C6D7
$C1B9-	LEDACTON_DV	$94	$C546
$C1BB-	LEDACTOFF_DV	$96	$C54F
$C1BD-	ERRLEDON_DV	$98	$C558
$C1BF-	ERRLEDOFF_DV	$9A	$C561
$C1C1-	SIDE_DV	$9C	$C56A
$C1C3-	BUFMOVE_DV	$9E	$C589
$C1C5-	WRTVER_DV	$A0	$C9E1
$C1C7-	TRKWRT_DV	$A2	$C5AC
$C1C9-	SP_READ	$A4	$C800
$C1CB-	SP_WRITE	$A6	$C700
$C1CD-	PSEEK_DV	$A8	$C6D7
$C1CF-	TREAD_DV	$AA	$CB09
$C1D1-	TWRT_DV	$AC	$CAE4
$C1D3-	SEEKHD_DV	$B0	$CB0F
$C1D5-	TPREAD_DV	$B2	$CB26
$C1D7-	TPWRT_DV	$B4	$CB26
$C1D9-	DETWP_DV	$B6	$CB35
$C1DB-	SEEKPHD_DV	$B8	$C900
$C1DD-	RESTORE_DV	$C0	$C900
$C1DF-	JUMPC_DV	$D0	$C900
$C1E1-	EXBUF_DV	$E0	$C900
$C1E3-	FORMATDK_DV	$F0	$CB76

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL NAME	DESCRIPTION COMMAND NUMBER
$C1E5-	CTRLERR_DV	NONE $CB85
$C1E7-$C2E6	DATABYTES	EIGHT DATA BYTES FOR EACH OF THE CONTROLLER COMMANDS LISTED ABOVE/JOB INDEX FOR 33 JOBS
$C2E7-	JOBRESET	RESET COMMAND - RESETS THE DISK CONTROLLER AND VARIABLES ($82)
$C30C-	SETWDCMDS	RESTORE DEFAULT WD177X COMMMAND TABLE
$C390-	JOBMOTON	MOTOR ON COMMAND- TURNS ON THE DRIVE SPINDLE MOTOR ($84)
$C393-	JOBMOTOFF	MOTOR OFF COMMAND- TURNS OFF THE DRIVE SPINDLE MOTOR ($86)
$C396-	JOBMOTONI	MOTOR ON IMMEDIATELY ($88)
$C3A9-	JOBMOTOFFI	MOTOR OFF IMMEDIATELY ($8A)
$C3AF-	JOBSEEKTRK	SEEKS TRACK COMMAND ($8C)
$C3BB-	JOBFORMATTRK	FORMAT ONE PHYSICAL TRACK ($8E)
$C3EC-	WRTINDEX	WRITE TRACK INDEX/SAVE AFTER INDEX HOLE
$C52C-	STOP1TRKFORMAT	TERMINATE PHYSICAL TRACK FORMAT
$C546-	JOBACTLEDON	TURN ON DISK ACTIVITY LED ($94)
$C54F-	JOBACTLEDOFF	TURN OFF DISK ACTIVITY LED ($96)
$C558-	JOBERRLEDON	TURN ON DISK ERROR LED ($98)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$C561-	JOBERRLEDOFF	TURN OFF DISK ERROR LED ($9A)
$C56A-	JOBSETSIDE	SET UP SIDE SELECT ELECTRONICS TO THE VALUE IN THE SIDES TABLE. ($9C)
$C589-	JOBMOVEDATA	MOVE DATA BETWEEN THE JOB QUEUE BUFFERS AND TRACK CACHE ($9E)
$C5AC-	JOBDUMPCACHE	DUMPS TRACK CACHE TO DISK (IF "DIRTY") ($A2)
$C5AF-$C5FF		NOT USED BY DOS
$C600-	DUMPOLD	DUMP OLD TRACK CACHE DATA
$C6D7-	DUALJOB	CHECKS TO SEE IF A DISK IS IN THE DRIVE AND SEEKS A PRESET PHYSICAL TRACK ($92 AND $A8)
$C6DD-$C6FF		NOT USED BY DOS
$C700-	JOBWRTPHYS	WRITE A PHYSICAL SECTOR DIRECTLY ($A6)
$C765-$C7FF		NOT USED BY DOS
$C800-	JOBREADPHYS	READ A PHYSICAL SECTOR DIRECTLY ($A4)
$C865-$C8FF		NOT USED BY DOS
$C900-	MULTIJOB	EXECUTES CONTROLLER COMMANDS FOR: READING SECTORS ($80) WRITING SECTORS ($90) SEEKING HEADERS ($B8) BUMP TO TRACK 0 ($C0) EXECUTE PROGRAM ($D0) EXECUTE BUFFER ($E0)

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$C9E1-	JOBVERCACHE	VERIFY CACHE DATA AGAINST A LOGICAL TRACKS DATA ($A0)
$C9F6-$C9FF		NOT USED BY DOS
$CA00-	FORMATVER	VERIFY DISK FORMAT
$CAE4-	JOBWRTLOG	WRITE A LOGICAL ADDRESS WITHOUT TRANSFER FROM JOB QUEUE BUFFER ($AC)
$CB09-	JOBREADLOG	READ A LOGICAL ADDRESS WITHOUT TRANSFER FROM JOB QUEUE BUFFER ($AA)
$CB0F-	JOBREADHDR	READ HEADER DATA FROM FIRST DISK SECTOR FOUND ($B0)
$CB26-	JOBWRTPHYS	WRITE A PHYSICAL ADDRESS WITHOUT TRANSFER FROM JOB QUEUE BUFFER ($B2)
$CB26-	JOBREADPHYS	READ A PHYSICAL ADDRESS WITHOUT TRANSFER FROM JOB QUEUE BUFFER ($B4)
$CB35-	JOBWRTPROTECT	CHECKS TO SEE IF THE CURRENT DISK IS WRITE PROTECTED ($00 = NO, $08 = YES) ($B6)
$CB76-	JOBFORMATDSK	FORMAT THE DISK WITH THE DEFAULT PHYSICAL FORMAT ($F0)
$CBB1-	MOTORON	SPINDLE MOTOR ON
$CBBA-	MOTOROFF	SPINDLE MOTOR OFF
$CBC3-	ACTLEDOFF	GREEN ACTIVITY LED OFF
$CBCC-	ACTLEDON	GREEN ACTIVITY LED ON
$CBEC-	WDCMDWAIT	WAIT UNTIL CURRENT COMMAND ON WD177X IS DONE

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$CBF4-	EXECWDCMD	EXECUTE WD177X COMMAND
$CC02-$CCFF		NOT USED BY DOS
$CD00-	FINDHDR	SEEK A DISK HEADER
$CD3F-	WDSTATUS	GET WD177X CONTROLLER STATUS
$CD5A-$CD62	WDSTDATA	VALUES $00, $05, $02 $00, $00, $00 $00, $00, $08
$CD63-	STARTMOT	FIRE UP THE DISK DRIVE MOTOR
$CD7B-	UPTOSPEED	WAIT TILL THE MOTOR IS UP TO SPEED
$CDBC-	DRVRDY	CHECK FOR DRIVE READY
$CEDC-	LOG2PHYS	LOGICAL TO PHYSICAL SECTOR TRANSLATION ROUTINE
$CE71-	STEPSET	ROUTINE FOR HEAD STEPPING AND SETTLING
$CFD1-	FORCEWDIRQ	FORCE INTERRUPT ON WD177X, DO A DELAY, WAIT TILL COMMAND IS DONE
$D00D-$D03D	LAZY1	FIRST DOS "LAZY" MESSAGE "AM I LAZY???...NO JUST WANTED TO SAVE A FEW MS..."
$D03E-$D53D	NOTHING1	COPIES WHAT POINTED TO BY $48 TO WHATS POINTED TO BY $4A INDEXED BY .YR
$D549-$D557	LAZY2	SECOND AND LAST DOS "LAZY" MESSAGE "THIS IS LAZY!!!"
$D558-$DA57	NOTHING2	SEE LOCATION $D03E

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$DA63-	CHKSUM	ROUTINE TO DO THE CYCLIC REDUNDANCY CHECKSUM
$DAFD-	IRQRTN	1581 IRQ ROUTINE
$DB36-$DB75	USERADDR	ADDRESSES OF COMMAND ROUTINES FOR USER0
$DB36-		$BB11 - BURST READ SECTOR
$DB38-		$BAFA - DRIVE NOT READY
$DB3A-		$BC01 - BURST WRITE SECTOR
$DB3C-		$BBF9 - DRIVE NOT READY
$DB3E-		$BCB2 - BURST READ SECTOR HEADER
$DB40-		$BAFA - DRIVE NOT READY
$DB42-		$BD12 - BURST FORMAT DISKETTE
$DB44-		$BD12 - BURST FORMAT DISKETTE
$DB46-		$BDFC - SYNTAX ERROR #31
$DB48-		$BDFC - SYNTAX ERROR #31
$DB4A-		$BE06 - BURST DETERMINE SECTOR SEQUENCE
$DB4C-		$BAFA - DRIVE NOT READY
$DB4E-		$BEBB - BURST INQUIRE STATUS
$DB50-		$BAFA - DRIVE NOT READY
$DB52-		$BEF8 - SYNTAX ERROR #31
$DB54-		$BEF8 - SYNTAX ERROR #31
$DB56-		$BB11 - BURST READ SECTOR
$DB58-		$BAFA - DRIVE NOT READY
$DB5A-		$BC01 - BURST WRITE SECTOR
$DB5C-		$BBF9 - DRIVE NOT READY
$DB5E-		$BCB2 - BURST READ SECTOR HEADER
$DB60-		$BAFA - DRIVE NOT READY
$DB62-		$BD12 - BURST FORMAT DISKETTE
$DB64-		$BD12 - BURST FORMAT DISKETTE
$DB66-		$89CB - RTS INSTRUCTION NO FUNCTION
$DB68-		$89CB - RTS INSTRUCTION NO FUNCTION
$DB6A-		$BE06 - BURST DETERMINE SECTOR SEQUENCE
$DB6C-		$BAFA - DRIVE NOT READY
$DB6E-		$BF02 - READ NEXT SECTOR HEADER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$DB70-		$BF02 - READ NEXT SECTOR HEADER
$DB72-		$AA3C - EXECUTE 1581 STATUS COMMAND
$DB74-		$B8D5 - FAST LOAD A FILE OVER THE 1581 BUS
$DB76-	BAMUSE	NUMBER OF BYTES IN BAM FOR EACH DISK TRACK ($06/6)
$DB77-	NAMEOFFSET	DISK NAME OFFSET IN BAM ($04/4)
$DB78-$DB83	DOSCMDTABLE	TABLE OF DOS COMMANDS V, I, /, M, B, U, P, &, C, R, S, N
$DB84-$DB8F	DOSCMDLO	DOS COMMAND VECTORS LOW BYTES
$DB90-$DB9B	DOSCMDHI	DOS COMMAND VECTORS HIGH BYTES V - $FF09 I - $FF0C / - $FF0F M - $FF12 B - $FF15 U - $FF18 P - $FF1B & - $FF1E C - $FF21 R - $FF24 S - $FF27 N - $FF2A
$DB9C-$DBA0	CMDIMAGES	STRUCTURE IMAGES FOR DOS COMMANDS $51 - DISK COPY $DD - RENAME A FILE (NOT PARSED) $1C - SCRATCH A FILE (NOT PARSED) $9E - FORMAT A DISK (NOT PARSED) $1C - LOAD A FILE

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$DBA1-$DBA4	FILEMODE	MODE TABLE $DBA1- $52 R = READ MODE $DBA2- $57 W = WRITE MODE $DBA3- $41 A = APPEND MODE $DBA4- $4D M = MODIFY MODE (READ AN IMPROPERLY CLOSED FILE)
$DBA5-$DBBC	FILETYPE	FILE TYPE TABLE
	FT0	$DBA5-$DBAA FIRST BYTE OF FILE TYPE (D, S, P, U, L, C) FOR FILE OPERATIONS
	FT1	$DBAB-$DBB0 FIRST BYTE OF FILE TYPE (D, S, P, U, R, C)
	FT2	$DBB1-$DBB6 SECOND BYTE OF FILE TYPE (E, E, R, S, E, B)
	FT3	$DBB7-$DBBC THIRD BYTE OF FILE TYPE (L, Q, G, R, L, M) VALID FILE TYPES ARE: DEL, SEQ, PRG, USR, REL, CBM
$DBBD-$DBC1	ERRFLAG	ERROR FLAG VARIABLES FOR USE BY BIT COMMANDS
$DBC2-$DBC6	ERROFFSETS	OFFSETS FOR ERROR RECOVERY
$DBC7-	SPINPATCH	SPIN ROUTINE PATCH TO CLEAR THE SHIFT REGISTER
$DBE0-	SPOUTPATCH	SPINOUT ROUTINE PATCH TO CLEAR THE SHIFT REGISTER

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$DBEE-	RELEASESEC	RELEASE SECTORS IN BAM AFTER SCRATCHING A FILE/USED AFTER SCRATCHING A PARTITION FILE TO UPDATE THE DISK BAM
$DBF4-	SEEKNCHK	SEEK A HEADER AND CHECK DISK FORMAT
$DC01-$DC37	CRMSG	COMMODORE DOS COPYRIGHT MESSAGE "(C)1987 COMMODORE ELECTRONICS LTD., ALL RIGHTS RESERVED"
$DC38-$FEFF		NOT USED BY THE DOS
$FF00-	IDLE	EXECUTE THE JIDLE ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $0190. JIDLE AT $B0F0
$FF03-	IRQ	EXECUTE THE JIRQ ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $0192. JIRQ AT $DAF0
$FF06-	NMI	EXECUTE THE JNMI ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $0194. JNMI AT $AFCA
$FF09-	VERDIR	EXECUTE THE JVERDIR ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $0196. JVERDIR AT $B262
$FF0C-	INTDRV	EXECUTE THE JINTDRV ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $0198. JINTDRV AT $8EC5

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$FF0F-	PART	EXECUTE THE JPART ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $019A. JPART AT $B781
$FF12-	MEM	EXECUTE THE JMEM ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $019C. JMEM AT $892F
$FF15-	BLOCK	EXECUTE THE JBLOCK ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $019E. JBLOCK AT $8A5D
$FF18-	USERVEC	EXECUTE THE JUSER ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01A0. JUSER AT $898F
$FF1B-	RECORD	EXECUTE THE JRECORD ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01A2. JRECORD AT $A1A1
$FF1E-	UTLODR	EXECUTE THE JUTLODR ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01A4. JUTLODR AT $A956
$FF21-	DSKCOPY	EXECUTE THE JDSKCPY ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01A6. JDSKCPY AT $876E
$FF24-	RENAMEVEC	EXECUTE THE JRENAME ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01A8. JRENAME AT $88C5

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$FF27-	SCRTCH	EXECUTE THE JSCRTCH ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01AA. JSCRTCH AT $8688
$FF2A-	NEW	EXECUTE THE JNEW ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01AC. JNEW AT $B348
$FF2D-	ERROR	EXECUTE THE ERROR ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01AE. ERROR AT $A7AE
$FF30-	ATNSERV	EXECUTE THE JATNSRV ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01B0. JATNSRV AT $ABCF
$FF33-	TALK	EXECUTE THE JTALK ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01B2. JTALK AT $AD5C
$FF36-	LISTEN	EXECUTE THE JLISTEN ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01B4. JLISTEN AT $AEB8
$FF39-	LCC	EXECUTE THE JLCC ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01B6. JLCC AT $C0BE
$FF3C-	TRANSTS	EXECUTE THE JTRANS_TS ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01B8. JTRANS_TS AT $CEDC

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$FF3F-	CMDERR	EXECUTE THE CMDERR ROUTINE VIA AN INDIRECT JUMP TO THE VECTOR AT $01BA. CMDERR AT $A7F1
$FF42-$FF53		NOT USED BY DOS
$FF54-	STROBECTRLER	EXECUTE THE JSTROBE_CTRLER ROUTINE VIA A DIRECT JUMP TO $FF54 JSTROBE_CTRLER AT $959D
$FF57-	CBMBOOT	EXECUTE THE JCBMBOOT ROUTINE VIA A DIRECT JUMP TO $FF57 JCBMBOOT AT $A938
$FF5A-	CBMBOOTRTN	EXECUTE THE JCBMBOOTRTN ROUTINE VIA A DIRECT JUMP TO $FF5A JCBMBOOTRTN AT $A94C
$FF5D-	SIGNATURE	EXECUTE THE JSIGNATURE ROUTINE VIA A DIRECT JUMP TO $FF5D JSIGNATURE AT $AB1D
$FF60-	DEJAVU	EXECUTE THE JDEJAVU ROUTINE VIA A DIRECT JUMP TO $FF60 JDEJAVU AT $9145
$FF63-	SPINOUT	EXECUTE THE JSPINOUT ROUTINE VIA A DIRECT JUMP TO $FF63 JSPINOUT AT $AEEA
$FF66-	ALLOCBUFF	EXECUTE THE JALLOCBUFF ROUTINE VIA A DIRECT JUMP TO $FF66 JALLOCBUFF AT $8C5C
$FF69-	TESTTRKSEC	EXECUTE THE JTESTTRKSEC ROUTINE VIA A DIRECT JUMP TO $FF69 JTESTTRKSEC AT $9460

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$FF6C-	DUMPTRK	EXECUTE THE DUMP TRACK TO DISK ROUTINE AT $BFE3
$FF6F-$FF74		NOT USED BY DOS
$FF75-$FFA0	RAMVECDATA	VECTORS FOR RAM JUMP TABLE AT $0190 (DEFAULTS SEE $FF00-$FF3F ABOVE)
$FFA1-$FFAC		NOT USED BY DOS
$FFAD-	INITJMP	ROUTINE TO INITIALIZE RAM JUMP TABLE AT $0190
$FFC8-$FFE9		NOT USED BY DOS
$FFEA-$FFFF	USRJMP	USER COMMAND JUMP TABLE
$FFEA-	USER1	"U1/A" VECTOR POINTS TO $8B9A
$FFEC-	USER2	"U2/B" VECTOR POINTS TO $8BD7
$FFEE-	USER3	"U3/C" VECTOR POINTS TO $0500
$FFF0-	USER4	"U4/D" VECTOR POINTS TO $0503
$FFF2-	USER5	"U5/E" VECTOR POINTS TO $0506
$FFF4-	USER6	"U6/F" VECTOR POINTS TO $0509
$FFF6-	USER7	"U7/G" VECTOR POINTS TO $050C
$FFF8-	USER8	"U8/H" VECTOR POINTS TO $050F
$FFFA-	NNMI	"U9/I" NNMI ROUTINE POINTS TO $AD3C
$FFFC-	DSKINT	"U:/J" DSKINT ROUTINE POINTS TO $AF24

© Copyright 1988 D. W. Martin All Rights Reserved.

LOCATION	LABEL	DESCRIPTION
$FFFE-	SYSIRQ	"UK" SYSIRQ ROUTINE POINTS TO $FF03 (SENDING A "UK" COMMAND CRASHES THE DRIVE)

© Copyright 1988 D. W. Martin All Rights Reserved.

Chapter 9 - Miscellaneous

Power-up Diagnostics

The 1581 performs a series of self-tests on power-up of the RAM, ROM and Controller. If a failure is detected, the DOS will blink all the LED's a specific number of times. The flashing code repeats continuously.

Number of Flashes	Resource	Component
1	Zero Page	8K x 8 RAM
2	ROM	25256 ROM
3	$1000-$1FFF	8K x 8 RAM

An additional test performed by the 1581 is the controller test. If a controller failure is encountered the error channel will contain "76, CONTROLLER ERROR,00,00. "

If an extensive test of the ROM is needed, the 1581 provides a signature analysis via the command "U0>T. " If a failure is detected the LED's will blink 4 times continuously.

* * *

TRACK CACHE BUFFER

The 1581 disk drive has two read/write heads, 80 disk cylinders, ten 512 byte sectors per cylinder and 10,240 usable data bytes per cylinder. Although the disk is physically organized in 512 byte sectors, the DOS requests data in 256 byte chunks. The controller interprets the DOS sector request and translates the logical request to a physical request on the diskette. The track and sector translation routine may be revectored by the user (This feature might make custom formats possible).

All I/O to the diskette is implemented as an entire track access. This allows access to the drive with no interleaving. When a sector is read the drive has to only copy the data from the track cache buffer to one of the job queue buffers. If the requested sector appears on another track then that track is read into the cache buffer. Note, that if the track cache buffer contains data from a previously read track and some of that data has been changed, then the old data is written back to the diskette before

© Copyright 1988 D. W. Martin All Rights Reserved.

the new data is read in.

Dumping of the track cache buffer is automatic, but it can be manually controlled. The track cache is dumped if the next physical request is for another track, a dump track buffer command is executed, or after 250 ms of idle serial bus activity.

* * *

1581 SERIAL BUS SPECIFICATIONS

The Commodore 1581 serial bus supports standard, fast and fast serial communications just like the 1571.

A Host Request Fast (HRF) command places the drive in fast serial mode. The 1581 remains in fast serial mode until it receives one of the following: Unlisten, Untalk, or a serial bus error. The HRF command generates a message that lets the host know that the addressed peripheral can receive bytes fast or slow.

The heart of the 1581's serial communication abilities is the 8520A 2 mhz CIA chip. This chip contains a programmable baud rate generator which is used for fast serial transfers on the 1581.

Timer A of the 8520A is used for the baud rate generator. In the output mode data is shifted out on the serial bus pin five (data) at half the underflow rate of Timer A and pin five will go low. The maximum baud rate possible is phi two divided by four, but the maximum usable baud rate will be determined by line loading and the speed at which the receiver responds to the input data.

Transmission of data will start following a write to the Serial Data Register (only if Timer A is running and in continuous mode). The clock derived from Timer A appears on the serial bus pin four (clock). The data in the Serial Data Register will be loaded into the shift register then shifted out to serial bus pin five (data).

After eight pulses on serial bus pin four (clock), a bit is set in the Interrupt Control Register and if desired an interrupt may be generated. Note, that all incoming data bytes cause an interrupt in the fast serial drive and that bytes are shifted out with the most significant bit first.

© Copyright 1988 D. W. Martin All Rights Reserved.

The following table lists the possible baud rates:

BAUD RATE	HI TIMER VALUE	LOW TIMER VALUE
166K	00	06
143K	00	07
100K	00	10
50K	00	20
25K	00	40
12.5K	00	80

NOTE: Values are clocked at a 2 mhz clock speed.

* * *

SERIAL BUS PIN OUT

PIN NUMBER	NAME	DESCRIPTION
1	SRQ	SERVICE REQUEST. FAST SERIAL WILL USE THIS LINE AS A TWO DIRECTION FAST CLOCK LINE.
2	GND	GROUND CONNECTION
3	ATN	ATTENTION SIGNAL. IDENTIFIES CONTROLLER COMMANDS. NO RESPONSE YIELDS DEVICE NOT PRESENT ERROR (IN ONLY)
4	CLOCK	USED FOR TIMING THE DATA SENT ON THE SERIAL BUS (SOFTWARE CLOCKED) (IN AND OUT)
5	DATA	DATA ON THE SERIAL BUS IS SENT ON BIT AT A TIME (SOFTWARE TOGGLED). THIS LINE IS WIRE "ORED" AND USED AS THE FAST DATA LINE TO COMPLIMENT THE FAST CLOCK ON THE SRQ LINE (IN AND OUT)
6	RESET	RESETS THE PERIPHERAL UPON HOST RESET

NOTE: The signal on pin 3 goes low when an ATN command is found and an interrupt is generated. The attention sequence is followed by an address or command.

© Copyright 1988 D. W. Martin All Rights Reserved.

SERIAL BUS COMMANDS/ADDRESSES

The following table lists the commands (addresses) and gives their explanation:

COMMAND NAME	LABEL	BINARY VALUE
HOST REQUEST FAST	HRF	1111 1111
DEVICE REQUEST FAST	DRF	0000 0000
TALK ADDRESS	TA	010X XXXX
LISTEN ADDRESS	LA	001X XXXX
UNTALK	UNTLK	0101 1111
UNLISTEN	UNLSN	0011 1111
SECONDARY ADDR OPEN	SAO	1111 YYYY
SECONDARY ADDR CLOSE	SAC	1110 YYYY
SECONDARY ADDR NORMAL	SA	011Z ZZZZ

NOTE: The HRF and DRF use a fast byte clocked over the SRQ line.

Device addresses (X XXXX above) for talk and listen can hold the following values:

ADDRESSES	FUNCTION
0 TO 3	INTERNAL DEVICES
4 TO 7	NORMAL CBM PRINTERS
8 TO 11	NORMAL DISK UNITS
12 TO 30	UNUSED

Channel addresses (YYYY above) for open and close can hold the following values:

ADDRESSES	FUNCTION
0	PROGRAM TYPE READ DATA CHANNEL
1	PROGRAM TYPE WRITE DATA CHANNEL
2 TO 14	CHANNEL FOR ALL FILE TYPES (R/W)

© Copyright 1988 D. W. Martin All Rights Reserved.

ADDRESSES	FUNCTION
15	COMMAND CHANNEL (R/W)

NOTE: R/W means read/write.

* * *

SECONDARY ADDRESS DESCRIPTIONS

ADDRESS	DESCRIPTION
0	LOAD CHANNEL (LOADING FILES)
1	SAVE CHANNEL (SAVING FILES)
2 TO 14	OPEN A BUFFER OR BUFFERS DIRECT BUFFER ACCESS OR FILE ACCESS. EIGHT BUFFERS ARE AVAILABLE.
15	COMMAND AND ERROR CHANNEL (SEND COMMANDS TO THE DRIVE OR READ THE ERROR CHANNEL)

The normal secondary address (Z ZZZZ above) can be from zero to 31.

* * *

STANDARD KERNAL CALLS AND SERIAL BUS

The following are a list of standard kernal calls and how they communicate with the 1581 serial bus. A few new terms have been added they are defined below to help you understand how the Kernal calls work.

TERM	DESCRIPTION
DB	DATA BYTE
FN	FILE NAME BYTE
EOI	END OR IDENTIFY HANDSHAKE
TKATN	TALK/ATN HANDSHAKE

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

LOAD FROM DISK - KERNAL LOAD

This routine loads data bytes from any input device and places it directly into the hosts memory.

```
HRF LA SAO FN1 FN2...FNn-1 EOI FNn HRF UNLSN HRF
 TA SA

TKATN DB1 DB2...DBn-1 EOI DBn UNTLK HRF TA SAC
 HRF UNLSN
```

* * *

SAVE TO DISK - KERNAL SAVE

This routine saves a section of memory from the host to a diskette.

```
HRF LA SAO FN1 FN2...FNn-1 EOI FNn HRF UNLSN HRF
 LA SA

DRF DB1 DB2...DBn EOI DBn-1 HRF UNLSN HRF LA SAC
 HRF

UNLSN
```

* * *

OPEN WITH SECONDARY ADDRESS - KERNAL OPEN

This routine is used to open a logical file for I/O operations.

```
HRF LA SAO DRF FN1 FN2...FNn-1 EOI FNn HRF UNLSN
```

* * *

CLOSE WITH SECONDARY ADDRESS - KERNAL CLOSE

This routine is used to close a logical file after all I/O operations have been completed on that file.

```
HRF LA SAC HRF UNLSN
```

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

OPEN A CHANNEL FOR OUTPUT - KERNAL CHKOUT

This routine must be called before any data is sent to any output device.

HRF LA SA

* * *

OPEN A CHANNEL FOR INPUT - KERNAL CHKIN

This routine is called to define any previously opened channel as an input channel.

HRF TA SA TLKATN

* * *

OUTPUT A CHARACTER - KERNAL CHROUT

This uses a single character buffer and will send previously buffered characters if any exists. This buffer is also sent along with EOI prior to sending any serial bus command sequence (HRF, LA, TA, SAO, SAC, SA, UNTLK, UNLSN).

* * *

INPUT A CHARACTER - KERNAL CHKIN/GETIN

This routine is called to get a byte of data from a channel already setup as an input channel.

DBc or EOI DBc (if the external device sends an EOI)

* * *

CLEAR I/O CHANNELS - KERNAL CLRCHN/CLRCHN

This routine is used to clear and restore all open channels to their default values.

If a CHKIN channel open was used: UNTLK

If a CHKOUT channel open was used: EOI DBc HRF UNLSN

* * *

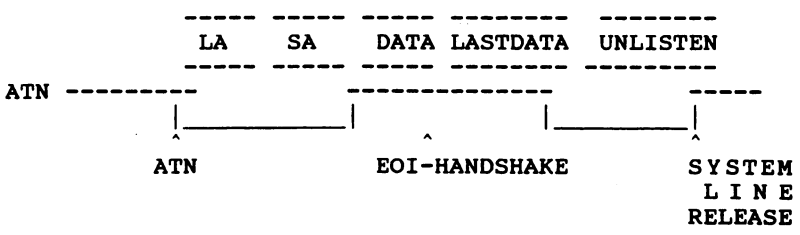
© Copyright 1988 D. W. Martin All Rights Reserved.

CHECK FOR STOP KEY PRESSED - KERNAL STOP

This routine scans to see if the stop key is pressed. If the stop key is pressed the routine calls CLRCHN.

* * *

STANDARD SERIAL BUS PROTOCOL



The data is processed by synchronous transfer with pin 4 (CLOCK) on the serial bus. The first bit is the LSB and all bits are of 8 bit construction.

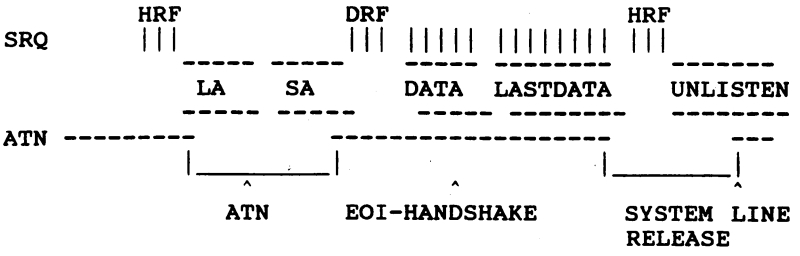
The talk and listen addresses are $48 and $28 for device eight or $49 and $29 for device nine.

The secondary address (SA) is $6x , $Fx, or $Ex (where x has a value from $00 to $0F).

Speed: 4,800 to 6,800 baud.

* * *

FAST SERIAL BUS PROTOCOL



The data is processed by synchronous transfer with pin 1 (SRQ) on the serial bus. The first bit is the MSB and all bits are of 8 bit construction. The minimum bit cell is 6 us. or 48 us. per byte.

© Copyright 1988 D. W. Martin All Rights Reserved.

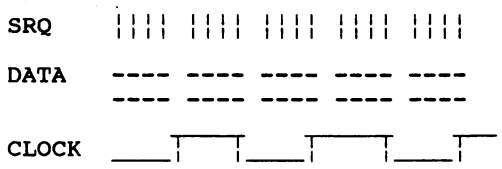
The talk and listen addresses are $48 and $28 for device eight or $49 and $29 for device nine.

The secondary address (SA) is $6x , $Fx, or $Ex (where x has a value from $00 to $0F).

Speed: 21,000 to 23,000 baud.

* * *

BURST SERIAL BUS PROTOCOL



The data is processed by synchronous transfer with pin 1 (SRQ) on the serial bus. The first bit is the MSB and all bits are of 8 bit construction. The minimum bit cell is 6 us. or 48 us. per byte. Handshaking is processed by pin 4 (CLOCK).

Speed: 78,400 to 80,000 baud.

* * *

1581 Bugs

Soon after it's appearance on the market the 1581 disk drive was plagued with several unfortunate hardware problems that Commodore has since corrected.

Extensive testing of the 1581 was done and three manufacturing problems were corrected.

The first one involved an improperly grounded jumper (J1) on the PC board inside the 1581 disk drive. Allows the drive to use the 6 ms step rate.

The second involved a bad batch of WD1770 chips that were later replaced with the WD1772 chip. A typical symptom of this problem is a corrupted directory after a disk copy.

The third involves a bad solder connection of pin 10 on device U10. It causes intermitant "device not present" errors.

© Copyright 1988 D. W. Martin All Rights Reserved.

Any authorized Commodore repair center can repair a 1581 that suffers any of the above problems. I'm not positive, but I think they will do these repairs whether your drive is in warranty or not. Contact your repair center and ask for details on repairing the drive and about warranty coverage.

You can test your drive with a public domain diagnostic program called 1581.DIAG or you can open the drive up and inspect it's insides yourself. The 1581.DIAG program was written by a concerned employee at Commodore. You can probably find this file on your local BBS and it's definitely circulating on most of the major telecommunication newtworks. The original program requires that you own a C128 since it contains BASIC 7.0 commands. However, since not everyone out there owns a C128 I have included a public domain version of the diagnostic program that will run on both computers. You'll find the program on the companion disk called 1581.DIAG.C64. This file is provided free of charge as a service to those that do not have access to modems or user groups. Feel free to share 1581.DIAG.C64 with your friends.

Using the diagnostic program is easy. Simply load and run it from within BASIC 2.0. Now, insert a blank diskette into the drive to be tested and follow the on screen prompts.

The results you want to see are: "UNIT X CONTAINS A WD1772 AND J1 IS CLOSED. " Please note that the program cannot test the status of pin 10 on U10. I'm also told that having a WD1770 or WD1772 is ok, but if you are experiencing problems with a WD1770 switch it to the WD1772. The switch will have to be done by a qualified repairman since the WD177X controller is not socketed.

During my explorations of the 1581 disk drive I have found only one apparent error in the DOS. The routine at $BA64 sets the error vector at $01BA-$01BB to point at an unused portion of the ROM that contains $FF/255 in hex. The exact location is $DFDF. An attempt to execute machine code at that location will crash the drive.

Although I have crashed the drive many times while exploring the DOS. I have yet to find a time when the routine at $BA64 caused the drive to crash by resetting the error vector mentioned above.

© Copyright 1988 D. W. Martin All Rights Reserved.

In the meantime, no other errors have surfaced. As a third generation of DOS (1541 to 1571 to 1581) the DOS seems to be pretty secure and reliable. If you know of any problems or have documented a bug please share the information with the rest of us.

* * *

Reaching the Author

This book is the result of many hours of hard work that covers a period of nearly 12 months. During that time I have spent many hours researching the subjects covered in this book. If you find any errors or omissions in my memory maps or other parts of the book please feel free to contact me in care of:

Software Support International
2700 NE Andresen Road #A-1
Vancouver, WA, USA 98661

If you write and would like a reply please send an SASE.

If you own an MSD disk drive then you might be interested in my book The MSD DOS Reference Guide. It contains sections on using and programming the DOS, comprehensive RAM and I/O memory maps and a commented disassembly of the MSD DOS V2.3. The ONLY MSD DOS reference available. Comes complete with a companion disk that's full of programs for the MSD drives. Contact Software Support International for price and availability.

* * *

Getting Information on the WD177x

Official information on the WD177x series chips can be really handy if you plan on heavy programming of the 1581 disk drive. Or even if you would just like to learn more about the floppy disk controller chip then contact Western Digital at:

Western Digital Corporation
2445 McCabe Way
Irvine, California, USA 92714

Phone: (800)847-6181 or (714)863-0102

Ask for documentation on the WD1770/1772 5.25" Floppy Disk Controller/Formatter. I called and was

© Copyright 1988 D. W. Martin All Rights Reserved.

mailed the information FREE several months ago. I'm not sure if the free information policy still stands, but just write or call them and find out.

The documentation they sent me was kind of technical so the beginner may not find it as useful as more experienced people would.

* * *

RECOMMENDED READING

I suggest that beginners or experts in DOS programming read the following books. They are very informative and make great reference books.

Inside Commodore DOS by Richard Immers and Gerald G. Neufeld $19.95 (ISBN 0-8359- 3091-2)

1571 Internals by Rainer Ellinger $19.95 (ISBN 0-916439-44-5)

The Anatomy of the 1541 Disk Drive by Lothar Englisch and Norbert Szczepanowski $19.95 (ISBN 0-916439-01-1)

The MSD DOS Reference Guide by David W. Martin $29.95

Commodore 1581 Disk Drive User's Guide by Commodore Business Machines (P/N:319928-01) $199.95 (or free with drive purchase - hmmm maybe you could buy it separately?!?)

These books should be available through your local book retailer and some mail order companies.I recommend them all highly, since without them I would not have learned what I know today about Commodore disk drives.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

CONSTRUCTION

Products used to produce this book include:

HARDWARE

Commodore 128D Computer and 1581 Disk Drive (The best 8-bit set up ever put together.)

Commodore Amiga 500 Computer (True computing power and YES Amiga does make it possible!)

Hewlett-Packard DeskJet printer (Laser-quality output at an affordable price.)

Software

WordPerfect 4.1 for the Amiga from WordPerfect, Corp. (The best wordprocessor on the Amiga.)

Kracker-Mon from Kracker-Jax (For enjoyable disk drive explorations.)

Super SnapShot V3.0 from LMS, Technologies (Great piece of hardware/software. A must have item!)

The text in this book was created using WordPerfect on the Amiga. Software and disk explorations using the 1581 were completed on the C128D computer. The final draft of the book was created on the Amiga using the HP DeskJet printer.

Let the list above be a lesson to everyone. No matter what someone says about your computer never let them say it can't do everything. Frankly, it can.

Good software and hardware combinations like the ones listed above only prove how true this is.

Thanks...

Special thanks to the folks who made this book possible. You all were helpful in lighting up some of the darker corners within the 1581 DOS and it's inner workings. From the West coast to the East coast and North to Canada many thanks.

This book would also not have been possible without loud rock music, days off without homework,

© Copyright 1988 D. W. Martin All Rights Reserved.

my parents, my dog Gretel, and my brother Andy (Whose loud and abrasive editorial comments were well taken, but largely ignored.)

© Copyright 1988 D. W. Martin All Rights Reserved.

Appendix A - Sample and Utility Programs

Loading Sample and Utility Programs

Loading and executing the sample and utility programs on the companion disk is as easy as 1..2..3. Simply insert the companion diskette in any drive and type:

LOAD "PROGRAM NAME", DEV# (HIT RETURN)

DEV# is any valid drive device number from 8 to 11.

PROGRAM NAME is the name of the file on the disk that you wish to LOAD. File names for each of the sample and utility programs are given in their respective documentation.

Executing the programs is as easy as loading. If an error did not occur during loading then type the following after the next **READY.** message appears:

RUN (HIT RETURN)

The loaded program will now execute. See the documentation on the following pages for information on using the sample and utility programs.

Note: New users should thoroughly read the manuals that accompany their computer and disk drives. These books provide detailed information on loading and executing files.

* * *

Loading Kracker-Mon

Kracker-Mon can be loaded and executed very easily. Simply insert the companion diskette in any drive and type:

LOAD "KRACKER-MON/1581", DEV# (HIT RETURN)

After the program loads type:

RUN (HIT RETURN)

This procedure loads the main Kracker-mon program. The main program allows you to create different versions of the Kracker-Mon machine language monitor and edit the monitor's opcode table. See Chapter 5 for detailed instructions.

© Copyright 1988 D. W. Martin All Rights Reserved.

Using the monitors that the above program creates is just as easy as loading the main program above. Simply type:

LOAD "MONX000", DEV#, 1 (HIT RETURN)

The program will automatically execute after it loads.

* * *

Sample Program Documentation

1581 Diagnostic Program

PROGRAM NAME: 1581.DIAG.C64

This program will check your 1581 disk drive for some known hardware errors.

You will need a formatted diskette to use the program. Simply insert this diskette into the drive and when prompted enter the device number of the unit to be tested. Device numbers range from 8 to 11 normally. See chapter 9 for information on what the diagnostic messages mean.

* * *

DOS DUMP

PROGRAM NAME: DOS DUMP

This program will dump a copy of a disk drives ROM to a diskette. Both 1571 and 1581 drives are supported. You will need a blank diskette or one with about 200 blocks free. Simply enter the drive number you want the ROM download to occur on. The program will then download the DOS to a program called **DOS DUMP.1000** onto the disk present in the download drive.

The **DOS DUMP.1000** file loads into computer memory at $1000/4096. It can be explored using Kracker-Mon or a suitable disassembler.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

1581 File Lock/Unlock

PROGRAM NAME: 81FILELOCKON.OFF

This program allows you to lock files so that they may not be removed from the diskette using the scratch command.

To use the program, you must know the name of the file you wish to lock or unlock before running this program. You will be prompted to enter the name of the file to be changed and the drive number where it currently resides. Afterwards, the program will toggle the files lock status. Files that are not locked become locked and files that are locked become unlocked.

This program is an adaptation of a program in the public domain for the 1541 and 1571 disk drives. Original author unknown.

* * *

Program-2-USR File Convertor

PROGRAM NAME: PRG-2-USR

This program will convert single block program files to files compatible with the DOS '&' command format described in chapter 1. It can be used to create files that can be used with the '&' command and the autoboot feature on the 1581 disk drive.

The program will create USR files compatible with all Commodore drives that support the '&' command or autobooting file features.

Please note that your are limited to using a program that is just 252 bytes in length. However, clever programmers will use the DOS block-read commands to boot up any extra blocks that are necessary into the DOS data buffers.

* * *

Demo DOS '&' Command

PROGRAM NAME: DEMO & CMD

This program demonstrates the ability of the 1581 to execute a utility command file. It will prompt you for the drive containing the demo files and then proceed with the demo.

© Copyright 1988 D. W. Martin All Rights Reserved.

The demo causes the disk error LED to flash continuously for a few seconds. Afterwards, the demo turns off the error LED by stopping the flashing. The demo USR file is called: **FLASH LED.**

* * *

1581 Autoboot Files

PROGRAM NAME: AUTOBOOT DEMO

This program demonstrates the ability of the 1581 to execute an autobooting command file. It will prompt you for the drive containing the demo files and then proceed with the demo.

The demo resets the disk drive via the "UJ" command. Upon initialization, the 1581 finds a file called "COPYRIGHT CBM 86, " which it loads and executes within the DOS. The demo file causes the disk error LED to flash continuously for a few seconds. Afterwards, the demo turns off the error LED by stopping the flashing. Stopping the demo is accomplished by issuing an "I0" command (disk initialization).

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Appendix B - 1581 Error Creator

1581 Error Creator Description

The 1581 Error Creator is a program that allows 1581 users to create read errors on their 1581 diskettes. The errors that are created can then be used as part of a protection scheme or as a means of testing the Error Scanner program on this book's companion diskette.

LOADING THE ERROR CREATOR

Loading the Error Creator is easy. Simply insert the companion disk in any drive and type:

LOAD "81 ERROR CREATOR", DRV# (HIT RETURN)

DRV# is any legal device number from 8 to 11.

Now type:

RUN (HIT RETURN)

USING THE ERROR CREATOR

The following is a command summary that will help you to use the 1581 Error Creator.

COMMAND	DESCRIPTION
F	Format 1581 Diskette (Formats a range of tracks with the specified error - see below).
E	Select error type (None, 20, 23, or 27) Appendix C explains the error types.
B	Starting track (range 1 to 80) to format.
C	Ending track (range 1 to 80) to format.
D	New Drive (Changes the drive number and tells you what type of drive it is). Use after running the error creator to set the drive type.

© Copyright 1988 D. W. Martin All Rights Reserved.

COMMAND	DESCRIPTION
Q	Exit to BASIC (Does not confirm your exit request, but exits immediately after hitting "Q").
X	Send Disk command (ex. I0 will initialize the drive).
F1/F3	Displays a disk directory (SPACE pauses while STOP aborts)

* * *

NOTES ON THE 1581 ERROR CREATOR

The 1581 Error Creator is a great tool for learning and exploring disk protection schemes on the 1581. It's an advanced tool that must be used with caution.

As an advanced tool the 1581 Error Creator is not a toy, so be very careful when using it around important diskettes. It's a good idea to write protect diskettes containing important data, since once an error is created on a track the original data is not recoverable.

If you'd like to attempt to create other types of read errors then I suggest you study the drive code present in the Error Creator. Manipulating the data within can produce all kinds of results. Jump in with both feet and enjoy yourself if you discover a new and interesting use of this program please let me know.

The 1581 Error Creator was provided courtesy of KRACKER JAX, a division of Software Support International. 1581 Error Creator was written by Mike Howard. Thanks guys for another wonderful tool.

© Copyright 1988 D. W. Martin All Rights Reserved.

Appendix C - Programming the 1581 Job Queue

1581 Controller Job Queue

The 1581 Disk Drive User's Guide (UG) describes the DOS in ROM as a sort of split personality. This personality split is denoted by the two major components that make up the ROM DOS: the DOS and the Controller.

The DOS as a collective entity is a complex machine language program, that performs many important and complex functions. Such as communications with the host computer system and management of information storage on the diskette. Information management allows the DOS to create, modify and delete data contained in files on the disk.

The Controller is the part of the DOS that does not care about files as a whole. It deals with disk data on a very basic and physical level. It's only concern is for the reading and writing of individual sectors on the diskette.

The DOS and Controller communicate through what is known as the 1581 Job Queue. The Job Queue is a collection of memory registers in the drive that act as "mailboxes" according to the 1581 UG.

These "mailboxes" are scanned or polled periodically for Job Queue commands. The actual scanning or polling takes place every 10ms. According to the 1581 UG the timing is controlled by timer B on the 8520.

If a job is found in the Job Queue it will be executed and a value will be returned to the DOS. The returned value is the job's status byte. It will tell you if the job was successful or not.

The job queue can be accessed directly or via the DOS routine called JSTROBECTRLER.

Most Job Queue commands require parameters to operate correctly. The following pages will show you how to use each of the 1581's Controller Job command codes. This includes setting up the commands and example DOS programs.

The memory map below lists the important memory locations that a user needs to know if he or she plans to program the 1581's disk controller.

© Copyright 1988 D. W. Martin All Rights Reserved.

Job Queue Programmer's Memory Map

LOCATION IN HEX	DESCRIPTION
$0002 - $0008	Job Queue Controller commands for jobs 0 to 6. Each job uses 1 byte.
$0009 - $000A	Job Queue Controller commands for jobs 7 to 8. Used for BAM jobs only. Each job uses 1 byte.
$000B - $001C	Logical or physical track and sector for jobs 0 to 8 listed above. Each job uses 2 bytes.
$008B - $008C	Pointer to track cache buffer
$009F - $00A7	Contains the offset pointer that points into the track cache buffer for jobs 0 to 8 listed above. Each job uses 1 byte.
$01BC - $01CD	Translated (physical) track and sector for jobs 0 to 8 listed above. Each job uses 2 bytes.
$01CE - $01D6	Contains the physical side for each of the jobs 0 to 8 listed above. Each job uses 1 byte.
$0C00 - $1FFF	Track cache buffer

Note: See the RAM memory map in Chapter 6 for job queue buffer addresses and other interesting RAM memory locations.

* * *

Basic 1581 Controller Call

The basic controller call must contain the following steps:

1) Set up any parameters that are required by the job command being used. Such as track and sector information and/or disk side.

© Copyright 1988 D. W. Martin All Rights Reserved.

2) Write the job controller code into the appropriate job queue.

3) Wait for the job to finish it's execution.

4) Retrieve the job's status from the job queue.

The steps listed above present Controller programming in a nutshell. For the most part all the steps are important particularly step number one.

Step one involves setting up the disk Controller hardware to perform a specified operation on a particular track and sector or an entire track. It is very important to make sure that the information you use to set up the Controller electronics is correct. The Controller does not check the information given to it for errors (except illegal Controller commands). As far as track and sector parameters are concerned the Controller will do exactly what you tell it.

Therefore, it is important to stay within the allowable track and sector range when programming the DOS. Reading or writing beyond the allowable track range of 1 to 80 will cause unpredictable results and is not recommended. While accessing tracks past track 80 is not recommended it may be possible to access tracks as high as track 85 on the 1581.

Step two is performed after setting up the job queue. This is where you tell the drive what to do at the specified track and sector. Simply store a Controller command code in the job queue and watch the drive work.

Afterwards, steps three and four give you the status of the job being executed. If you load the value from the currently active job and bit seven is set to one then that job has not completed execution. When bit seven is reset to zero the job has completed it's execution. Loading the value from the previously active job gives you the command status.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

1581 Controller Status Byte

The command status is determined according to the following chart:

Status Code	Description
$00/$01	No error occurred
$02	Cannot find header block
$03	No address mark was detected
$04	Data block was not present
$05	CRC error found in data block
$06	Disk format error
$07	Disk verify error
$08	Write protect error (disk was write protected)
$09	CRC error found in header block
$0A	Usage reserved
$0B	Disk changed flag/Disk ID mismatch
$0C	Disk format was not logical
$0D	Floppy Disk controller error
$0E	Syntax error/invalid job number
$0F	No disk present in disk drive

* * *

1581 Job Controller Command Descriptions and Examples

The following sample job queue programming samples are located in job queue buffer number one at $0400. They use job queue buffer number zero ($0300) as a data buffer. The data read or written to the

© Copyright 1988 D. W. Martin All Rights Reserved.

disk involves the first sector of the directory track. Therefore do not run these routines with an important diskette in the drive. Use only a blank formatted diskette for experimenting.

Command Name: READ_DV

Command Code: $80

Command Description: Reads a logical sector into the job queue buffer.

Command Example:
```
0400  LDA #$28    ; TRK # 40
      LDX #$00    ; SEC # ZERO
      STA $0B     ; JOB 0 TRK PARAM
      STX $0C     ; JOB 0 SEC PARAM
      LDA #$80    ; JOB 0 COMMAND
      STA $02     ; JOB 0 COMMAND EXEC
LOOP  LDA $02     ; WAIT TIL JOBS DONE
      BMI LOOP
      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: WRTSD_DV

Command Code: $90

Command Description: Write the job queue buffer to the specified track and sector.

Command Example:
```
0400  LDA #$28    ; TRK # 40
      LDX #$00    ; SEC # ZERO
      STA $0B     ; JOB 0 TRK PARAM
      STX $0C     ; JOB 0 SEC PARAM
      LDA #$90    ; JOB 0 COMMAND
      STA $02     ; JOB 0 COMMAND EXEC
LOOP  LDA $02     ; WAIT TIL JOBS DONE
      BMI LOOP
      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: WRTVER_DV

Command Code: $A0

Command Description: Verify track cache buffer data against a logical track's data.

Command Example:
```
0400  LDA #$28    ; TRK # 40
      LDX #$00    ; SEC # ZERO
      STA $0B     ; JOB 0 TRK PARAM
      STX $0C     ; JOB 0 SEC PARAM
      LDA #$A0    ; JOB 0 COMMAND
      STA $02     ; JOB 0 COMMAND EXEC
LOOP  LDA $02     ; WAIT TIL JOBS DONE
      BMI LOOP
      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: SEEKHD_DV

Command Code: $B0

Command Description: Log in a disk by reading the information from the first header found. Does not update the track cache buffer.

Command Example:
```
0400  LDA #$B0    ; JOB 0 COMMAND
      STA $02     ; JOB 0 COMMAND EXEC
LOOP  LDA $02     ; WAIT TIL JOBS DONE
      BMI LOOP
      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: RESTORE_DV

Command Code: $C0

Command Description: Bumps the Read/Write head to track 0.

Command Example:
```
0400 LDA #$C0    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: JUMPC_DV

Command Code: $D0

Command Description: Executes machine language program code in the specified job queue buffer.

Command Example:
```
0400 LDA #$D0    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The code begins to execute at $0300 in the example above. The code to be executed must have been previously loaded into the job queue buffer.

* * *

Command Name: EXBUF_DV

Command Code: $E0

Command Description: Executes machine language program code in the specified job queue buffer, but waits until the drive motor is on and up to speed and the read/write head is in place.

© Copyright 1988 D. W. Martin All Rights Reserved.

```
Command Example: 0400 LDA #$E0    ; JOB 0 COMMAND
                      STA $02     ; JOB 0 COMMAND EXEC
                 LOOP LDA $02     ; WAIT TIL JOBS DONE
                      BMI LOOP
                      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The code begins to execute at $0300 in The example above. The code to be executed must have been previously loaded into the job queue buffer.

* * *

Command Name: RESET_DV

Command Code: $82

Command Description: Resets the disk drive Controller and RAM variables.

```
Command Example: 0400 LDA #$82    ; JOB 0 COMMAND
                      STA $02     ; JOB 0 COMMAND EXEC
                 LOOP LDA $02     ; WAIT TIL JOBS DONE
                      BMI LOOP
                      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: MOTON_DV

Command Code: $84

Command Description: Turns on the drive spindle motor and places a $01 in the job queue after the spin-up sequence is complete.

```
Command Example: 0400 LDA #$84    ; JOB 0 COMMAND
                      STA $02     ; JOB 0 COMMAND EXEC
                 LOOP LDA $02     ; WAIT TIL JOBS DONE
                      BMI LOOP
                      RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: MOTOFF_DV

Command Code: $86

Command Description: Turns the drive spindle motor off after the spin-down sequence is complete.

Command Example:
```
0400 LDA #$86    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: MOTONI_DV

Command Code: $88

Command Description: Spindle motor on immediately.

Command Example:
```
0400 LDA #$88    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: MOTOFFI_DV

Command Code: $8A

Command Description: Spindle motor off immediately.

Command Example:
```
0400 LDA #$8A    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: SEEK_DV

Command Code: $8C

Command Description: Looks for a particular physical track (or cylinder). The current physical track is in memory at $0027.

Command Example:

```
0400 LDA #$27    ; PHYSICAL TRK 27
     LDY #$00    ; SIDE 0
     STA $01BC   ; SET PHYSICAL TRK
     STY $01CE   ; SET DISK SIDE
     LDA #$8C    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: FORMAT_DV

Command Code: $8E

Command Description: Physical track format (formats half a cylinder) read/write head must be over the track to be formatted and the side must be selected.

Command Example:

```
     ; POSITION THE READ/WRITE HEAD

0400 LDA #$27    ; PHYSICAL TRK 27
     LDY #$00    ; SIDE 0
     STA $01BC   ; SET PHYSICAL TRK
     STY $01CE   ; SET DISK SIDE
     LDA #$8C    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP

     ; FORMAT THE PHYSICAL TRK CURRENTLY
     ; UNDER THE READ/WRITE HEAD

     LDA #$8E    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

© Copyright 1988 D. W. Martin All Rights Reserved.

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: DISKIN_DV

Command Code: $92

Command Description: Checks to see if a disk is inserted into the drive.

Command Example:
```
0400 LDA #$92    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. If the status byte is anything other then $00 or $01 then no disk is in the disk drive.

* * *

Command Name: LEDACTON_DV

Command Code: $94

Command Description: Turn on the drive activity LED.

Command Example:
```
0400 LDA #$94    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: LEDACTOFF_DV

Command Code: $96

Command Description: Turn off the drive activity LED.

Command Example:
```
0400 LDA #$96    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: ERRLEDON_DV

Command Code: $98

Command Description: Turn on the drive error LED (blinking).

Command Example:
```
0400 LDA #$98    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: ERRLEDOFF_DV

Command Code: $9A

Command Description: Turn off the drive error LED (not blinking).

Command Example:
```
0400 LDA #$9A    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

© Copyright 1988 D. W. Martin All Rights Reserved.

Command Name: SIDE_DV

Command Code: $9C

Command Description: Sets the side select electronics to the value specified in the current job queue side parameter.

Command Example:
```
0400 LDA #$00    ; SIDE 0
     STA $01CE   ; SET SIDE JOB 0
     LDA #$9C    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: BUFMOVE_DV

Command Code: $9E

Command Description: Moves data between the job queue buffers and the track cache buffer.

Command Example: Not available

NOTE: This command does not function as described in the 1581 User's Guide. Repeated attempts to use this command resulted in no action by the drive to copy anything anywhere. The User Guide's instructions on how to use this command are either faulty or the command itself does not work.

Substitute: The following machine language programs can be used to copy data to and from a job queue buffer and the track cache buffer.

```
; COPY DATA FROM JOB QUEUE BUFFER
; TO TRK CACHE BUFFER

0400 LDY #$00
LOOP LDA $0300,Y ; JOB 0 BUFFER
     STA $0C00,Y ; TRK CACHE
     INY
     CPY #$00
     BNE LOOP
```

© Copyright 1988 D. W. Martin All Rights Reserved.

```
                    RTS

                ; COPY DATA FROM THE TRK CACHE
                ; BUFFER TO THE JOB QUEUE BUFFER

                0400 LDY #$00
                LOOP LDA $0C00,Y ; TRK CACHE
                     STA $0300,Y ; JOB 0 BUFFER
                     INY
                     CPY #$00
                     BNE LOOP
                     RTS
```

* * *

Command Name: TRKWRT_DV

Command Code: $A2

Command Description: Dumps the track cache buffer to disk if the track cache modified flag is set.

Command Example:
```
0400 LDA #$A2    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: SP_READ

Command Code: $A4

Command Description: Reads the selected physical sector into drive RAM at $0300 in buffer number 0. The track cache is not used and the sector is always read from the disk.

Command Example:
```
0400 LDA #$27    ; PHYSICAL TRK 0
     LDX #$01    ; PHYSICAL SEC 0
     LDY #$00    ; SIDE 0
     STA $0B     ; SET PHYSICAL TRK
     STX $0C     ; SET PHYSICAL SEC
     STY $01CE   ; SET DISK SIDE
     LDA #$A4    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
```

© Copyright 1988 D. W. Martin All Rights Reserved.

```
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The contents of $0B/$0C are copied to $01BC/$01BD when the command is executed.

* * *

Command Name: SP_WRITE

Command Code: $A6

Command Description: Writes the selected physical sector from drive RAM at $0300 in buffer number 0. The track cache is not used.

Command Example:
```
0400 LDA #$27    ; PHYSICAL TRK 0
     LDX #$01    ; PHYSICAL SEC 0
     LDY #$00    ; SIDE 0
     STA $0B     ; SET PHYSICAL TRK
     STX $0C     ; SET PHYSICAL SEC
     STY $01CE   ; SET DISK SIDE
     LDA #$A6    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The contents of $0B/$0C are copied to $01BC/$01BD when the command is executed.

* * *

Command Name: PSEEK_DV

Command Code: $A8

Command Description: Seeks the selected physical track.

Command Example:
```
0400 LDA #$27    ; PHYSICAL TRK 0
     LDX #$01    ; PHYSICAL SEC 0
     LDY #$00    ; SIDE 0
     STA $0B     ; SET PHYSICAL TRK
     STX $0C     ; SET PHYSICAL SEC
     STY $01CE   ; SET DISK SIDE
     LDA #$A8    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
```

© Copyright 1988 D. W. Martin All Rights Reserved.

```
                    BMI LOOP
                    RTS          ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The contents of $0B/$0C are copied to $01BC/$01BD when the command is executed.

* * *

Command Name: TREAD_DV

Command Code: $AA

Command Description: Reads a logical address without transfer to the buffer used by the job queue.

Command Example:
```
0400 LDA #$28     ; LOGICAL TRK 28
     LDX #$01     ; LOGICAL SEC 0
     LDY #$00     ; SIDE 0
     STA $0B      ; SET LOGICAL TRK
     STX $0C      ; SET LOGICAL SEC
     STY $01CE    ; SET DISK SIDE
     LDA #$AA     ; JOB 0 COMMAND
     STA $02      ; JOB 0 COMMAND EXEC
LOOP LDA $02      ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS          ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: TWRT_DV

Command Code: $AC

Command Description: Writes a logical address without transfer to the buffer used by the job queue.

Command Example:
```
0400 LDA #$28     ; LOGICAL TRK 28
     LDX #$00     ; LOGICAL SEC 0
     LDY #$00     ; SIDE 0
     STA $0B      ; SET LOGICAL TRK
     STX $0C      ; SET LOGICAL SEC
     STY $01CE    ; SET DISK SIDE
     LDA #$AC     ; JOB 0 COMMAND
     STA $02      ; JOB 0 COMMAND EXEC
LOOP LDA $02      ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS          ; JOB COMPLETED
```

© Copyright 1988 D. W. Martin All Rights Reserved.

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Command Name: TPREAD_DV

Command Code: $B2

Command Description: Reads a physical address without transfer to the buffer used by the job queue.

Command Example:
```
0400 LDA #$27    ; PHYSICAL TRK 27
     LDX #$01    ; PHYSICAL SEC 1
     LDY #$00    ; SIDE 0
     STA $0B     ; SET PHYSICAL TRK
     STX $0C     ; SET PHYSICAL SEC
     STY $01CE   ; SET DISK SIDE
     LDA #$B2    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The contents of $0B/$0C are copied to $01BC/$01BD when the command is executed.

* * *

Command Name: TPWRT_DV

Command Code: $B4

Command Description: Writes a physical address without transfer to the buffer used by the job queue.

Command Example:
```
0400 LDA #$27    ; PHYSICAL TRK 27
     LDX #$01    ; PHYSICAL SEC 1
     LDY #$00    ; SIDE 0
     STA $0B     ; SET PHYSICAL TRK
     STX $0C     ; SET PHYSICAL SEC
     STY $01CE   ; SET DISK SIDE
     LDA #$B4    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The contents of $0B/$0C are

© Copyright 1988 D. W. Martin All Rights Reserved.

copied to $01BC/$01BD when the command is executed.

Command Name: DETWP_DV

Command Code: $B6

Command Description: Checks to see if the disk in the drive is write protected.

Command Example:
```
0400 LDA #$B6    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed. The status byte returns $08 is the disk is write protected, otherwise it returns a $00.

* * *

Command Name: FORMATDK_DV

Command Code: $F0

Command Description: Formats a diskette using the default physical format.

Command Example:
```
0400 LDA #$F0    ; JOB 0 COMMAND
     STA $02     ; JOB 0 COMMAND EXEC
LOOP LDA $02     ; WAIT TIL JOBS DONE
     BMI LOOP
     RTS         ; JOB COMPLETED
```

NOTE: The accumulator contains the status byte when the job is completed.

* * *

Another aspect of programming the 1581 disk drive is Commodore's "Kernal" jump table in the 1581 Operating System ROMs. The jump table provides a permanent bridge to the major DOS routines that will not change from one version of the DOS to the next.

Commodore recommends their use, since direct ROM calls may not be compatible with future DOS releases. The jump table provides a way of keeping programs compatible with the current version of the 1581 DOS and any future revisions. It is good programming practice to use the jump table.

© Copyright 1988 D. W. Martin All Rights Reserved.

DOS Kernal Jump Table

The jump table and it's functions are listed below:

NAME	LOCATION	DESCRIPTION
JIDLE	$FF00	Main idle loop. Waits for disk activity.
JIRQ	$FF03	DOS interrupt routine. An interrupt occurs when any of the following occur: ATN goes low, Fast serial byte is shifted in, timeout on a timer and the execution of a BRK instruction.
JNMI	$FF06	This command does a "soft" reset. Defaults are restored. Autoboot file is executed if present, RAM and ROM checksums are done, and the device number switches are read.
JVERDIR	$FF09	DOS VALIDATE command
JINTDRV	$FF0C	DOS INITIALIZE command
JPART	$FF0F	DOS PARTITION command
JMEM	$FF12	DOS MEMORY commands
JBLOCK	$FF15	DOS BLOCK commands
JUSER	$FF18	DOS USER commands
JRECORD	$FF1B	DOS RECORD command
JUTLLODR	$FF1E	DOS UTILITY LOADER command
JDSKCPY	$FF21	DOS COPY command
JRENAME	$FF24	DOS RENAME command

© Copyright 1988 D. W. Martin All Rights Reserved.

NAME	LOCATION	DESCRIPTION
JSCRTCH	$FF27	DOS SCRATCH command
JNEW	$FF2A	DOS NEW/FORMAT command
ERROR	$FF2D	DOS controller error handler
JATNSRV	$FF30	ATN server for the serial bus
JTALK	$FF33	Talk routine for the serial bus
JLISTEN	$FF36	Listen routine for the serial bus
JLCC	$FF39	DOS controller routine
JTRANS_TS	$FF3C	Translation routine for logical to physical sector translations
CMDERR	$FF3F	Handles DOS errors
JSTROBE_CTRLER	$FF54	Call the job controller directly
JCBMBOOT	$FF57	CBM Autoloader routine
JCBMBOOTRTN	$FF5A	Routine to return from the autoloader with the autoloader disabled (stops the autoloader from doing an endless repeat).
JSIGNATURE	$FF5D	ROM signature analysis routine
JDEJAVU	$FF60	Switch for autoloader boot on initialization or burst inquire/query. Calling this routine with the carry flag set turns the autoloader on. Clearing the carry flag turns the autoloader off.

© Copyright 1988 D. W. Martin All Rights Reserved.

JSPINOUT	$FF63	Sets up the fast serial direction as input or output. Calling this routine with the carry flag sets the serial direction as output. If the carry flag is cleared then the serial direction is set as input.
JALLOCBUFF	$FF66	Routine to allocate RAM buffers. Call this routine with the buffer number to allocate in the A register. (1 = buffer 0, etc.)
SETFORMAT	$FF69	Routine to setup variables for normal disk format.
DUMPTRK	$FF6C	Routine to dump the track cache to diskette.

© Copyright 1988 D. W. Martin All Rights Reserved.